Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Robert Stockwell on Thu, 01 Nov 2001 13:29:23 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:

> Glad you didnt look too carefully, as the test file I sent was wrong
> nyway - thought I had sent it *before* I started changing things, woops :(
> I noticed though that the Keyword had been converted to a string on your
> version.
>
> The correct file is now on my web site, sorry - clearly not enough coffee
> http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug.pro
> code below:
> http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug2.pro
> the below test though is *maybe* easier to follow
>
> Martin

AH, yes, the new code quite nicely destroys IDL 5.5 running on
Linux RedHat 7.
It seems to run just fine (i.e. does not crash)
on IDL 5.4 on Win2000 however. Interesting..

-bob

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Martin Downing on Thu, 01 Nov 2001 17:00:26 GMT
View Forum Message <> Reply to Message

Glad you didnt look too carefully, as the test file I sent was wrong
nyway - thought I had sent it *before* I started changing things, woops :(
I noticed though that the Keyword had been converted to a string on your
version.

The correct file is now on my web site, sorry - clearly not enough coffee
http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug.pro
code below:
http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug2.pro
the below test though is *maybe* easier to follow

Martin

```
;=====================================================
; demonstrates _REF_EXTRA 's keyword argument passing anomalies

; main test procedure
pro test_ref_extra_bug2, debug=debug
; set debug to give help output of passes extra parameters
    test_extra0, /key, debug=debug
    test_ref_extra2, /key, debug=debug
    ; THIS ONE MAY CRASH YOUR IDL SESSION
    test_ref_extra1, /key, debug=debug
end

pro test_extra0, debug=debug,_EXTRA=e
 print,    "TEST_EXTRA0"
 if keyword_set(debug) then  help,e,/st
   test_ref_extra1, debug=debug,_REF_EXTRA=e
end

pro test_ref_extra1, debug=debug,_REF_EXTRA=e
 print,    "TEST_REF_EXTRA1"
   if keyword_set(debug) then  help,e,/str; &   if defined(e) then print, e
   test_ref_extra2,debug=debug, _REF_EXTRA=e
end
pro test_ref_extra2, debug=debug,_REF_EXTRA=e
 print,    "TEST_REF_EXTRA2"
   if keyword_set(debug) then  help,e,/st ;&  if defined(e) then print, e
   test_extra1, _EXTRA=e
end

; the  final call
pro test_extra1,key1=key1, debug=debug,_EXTRA=e
 print,    "TEST_EXTRA1"
   if keyword_set(debug) then  help,e,/st
   if keyword_set(KEY1) then print, "KEY1 set - good!" else Print, "KEY1
unset"
end

==============================
```

Martin

>
> I haven't looked closely at this, but here are my results
> on linux idl v5.5:
>
> IDL> test_ref_extra_bug

> { x86 linux unix linux 5.5 Aug 28 2001 32 32}
> ;calling: test_extra, 3, /Key1
> TEST_EXTRA
> ** Structure <827ebdc>, 1 tags, length=2, data length=2, refs=1:
> KEY1 INT 1
> { 1}
> ;calling: test_ref_extra, 2, /Key1
> pro test_ref_ex,call, _REF_EXTRA=e
> E STRING = Array[1]
> pro test_ex,call, _EXTRA=e
> ** Structure <827ebdc>, 1 tags, length=2, data length=2, refs=1:
> KEY1 INT 1
> pro test_ref_ex,call, _REF_EXTRA=e
> ** Structure <827ebdc>, 1 tags, length=2, data length=2, refs=1:
> KEY1 INT 1
>
>
> Cheers,
> bob
>

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by John-David T. Smith on Thu, 01 Nov 2001 20:23:17 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
> Glad you didnt look too carefully, as the test file I sent was wrong
> nyway - thought I had sent it *before* I started changing things, woops :(
> I noticed though that the Keyword had been converted to a string on your
> version.
>
> The correct file is now on my web site, sorry - clearly not enough coffee
> http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug.pro
> code below:
> http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug2.pro
> the below test though is *maybe* easier to follow
>

It seems to me the point is being driven home that switching from
_REF_EXTRA to _EXTRA in the middle of the game is verboten: a _REF_EXTRA
stack should remain a _REF_EXTRA stack. Not sure I agree with the
method of their point-making.

Can you distill the problem to a simpler (and preferrably
non-pair-recursive) example?

JD

P.S. If you only use "_EXTRA" in your routine calls (as opposed to routine *definitions*), this bug vanishes. I have always recommended saving "_REF_EXTRA" only for the relevant function definitions which would like to return values through their inherited keywords, and using the simpler "_EXTRA" everywhere else.

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Mark Hadfield on Thu, 01 Nov 2001 20:28:40 GMT
View Forum Message <> Reply to Message

Well, I must confess that I haven't taken the time to work out what you guys are on about, because these keyword-inheritance things always confuse the hell out of me.

But I think I can add something to the discussion by noting that there *has* been a change (presumably intended) in the way _REF_EXTRA works between 5.4 and 5.5. In 5.4, the rules for precedence of keywords passed by reference were different from the rules for keywords passed by value; in 5.5 they are the same. There was a thread on this subject entitled "Keyword precedence" in August 2000, in which JD & I confused the hell out of everyone else & each other (well me, anyway). I also communicated with RSI Tech Support at about that time and tried to convince them to change the behaviour. They seem to have done so.

So, your bug may be my fault.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Martin Downing on Thu, 01 Nov 2001 23:30:29 GMT
View Forum Message <> Reply to Message

"JD Smith" <jdsmith@astro.cornell.edu> wrote in message news:3BE1AF35.26BD8C3C@astro.cornell.edu...
> Martin Downing wrote:

>>
>> Glad you didnt look too carefully, as the test file I sent was wrong
>> nyway - thought I had sent it *before* I started changing things,
woops :(
>> I noticed though that the Keyword had been converted to a string on your
>> version.
>>
>> The correct file is now on my web site, sorry - clearly not enough
coffee
>>   http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug.pro
>> code below:
>>   http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug2.pro
>> the below test though is *maybe* easier to follow
>>
>
> It seems to me the point is being driven home that switching from
> _REF_EXTRA to _EXTRA in the middle of the game is verboten: a _REF_EXTRA
> stack should remain a _REF_EXTRA stack. Not sure I agree with the
> method of their point-making.
>
> Can you distill the problem to a simpler (and preferrably
> non-pair-recursive) example?

Hi JD,

The second example was not recursive, but I agree its confusing!
Anyway I take the point to avoid switching from _ref_exta tp _extra
>
> JD
>
> P.S. If you only use "_EXTRA" in your routine calls (as opposed to
> routine *definitions*), this bug vanishes. I have always recommended
> saving "_REF_EXTRA" only for the relevant function definitions which
> would like to return values through their inherited keywords, and using
> the simpler "_EXTRA" everywhere else.

You are right again, this also cures the problem - I did not realise this
was a safe thing to do if it was defined as a _ref_extra routine.

Thanks

Martin

ps: The code is free to anyone who enjoys zapping PC-based IDL. Id be
interested to know if unix/mac is any more stable!

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Mark Hadfield on Fri, 02 Nov 2001 00:18:15 GMT
View Forum Message <> Reply to Message

From: "Martin Downing" <martin.downing@ntlworld.com>
> http://homepage.ntlworld.com/martin.downing/idl/test_ref_ext ra_bug2.pro

So the routine that's crashing your machine (and mine) is as follows
(shortened)...

```
pro test_ref_extra1,_REF_EXTRA=e
  test_ref_extra2, _REF_EXTRA=e    ; Here be dragons!
end
```

This isn't merely bad practice, it's wrong. The _REF_EXTRA keyword is only
used in routine definitions. When *calling* a routine, the acceptable forms
are "_EXTRA" and (as of 5.5) "_STRICT_EXTRA".

Still, it shouldn't crash IDL. It should be detected as a syntax error.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--
Posted from clam.niwa.cri.nz [202.36.29.1]
via Mailgate.ORG Server - http://www.Mailgate.ORG

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Martin Downing on Fri, 02 Nov 2001 09:59:58 GMT
View Forum Message <> Reply to Message

----- Original Message -----
From: "Mark Hadfield" <m.hadfield@niwa.cri.nz>
>
> So the routine that's crashing your machine (and mine) is as follows
> (shortened)...
>
> pro test_ref_extra1,_REF_EXTRA=e
> test_ref_extra2, _REF_EXTRA=e ; Here be dragons!
> end
>
> This isn't merely bad practice, it's wrong. The _REF_EXTRA keyword is only
> used in routine definitions. When *calling* a routine, the acceptable

forms
> are "_EXTRA" and (as of 5.5) "_STRICT_EXTRA".
>
> Still, it shouldn't crash IDL. It should be detected as a syntax error.
>
Hi Mark

It seemed logical to me that if arguments have to be passed through many preprocessing routines to a later routine, that the best way would be by reference to avoid copying (as would be the case if the Keyword was specified), if the final routine insists on a local copy (by value) why should that matter?
Well I guess I just didnt understand the manual (no change there!). Are you saying that this is determined purely by the definition, and that calling with _REF_EXTRA as below
> test_ref_extra2, _REF_EXTRA=e ; Here be dragons! (indeed)
is supposed to be a syntax error? That was just not clear to me from the help (and for that matter the programmer who wrote the syntax parser!!!)
I am concerned as to which causes passing by reference/value, the definition or the call?
This is almost as horrible as C++ inheritance simple in principle but boy can some strange things happen when things get complicated - for now I guess will have to stick to _extra which at least is easy to understand :(

roll on coffee time!

cheers

Martin

Help on Keyword Inheritance from 5.4 follows:

"Choosing a Keyword Inheritance Mechanism

The "pass by reference" (_REF_EXTRA) keyword inheritance mechanism was introduced in IDL version 5.1, and in many cases is a good choice even if values are not being passed back to the calling routine. Because the _REF_EXTRA mechanism does not create an IDL structure to hold the keyword/value pairs, overhead is slightly reduced. Two situations lend themselves to use of the _REF_EXTRA mechanism:

1. You need to pass the values of keyword variables back from a called routine to the calling routine.
2. Your routine is an "inner loop" routine that may be called many times. If the routine is called repeatedly, the savings resulting from not creating a new IDL structure with each call may be significant.

It is important to remember that if the routine that is passing the keyword values through also needs access to the values of the keywords for some reason, you must use the "pass by value" (_EXTRA) mechanism.

Note - Updating existing routines that use _EXTRA to use _REF_EXTRA is relatively easy. Since the called routine uses _EXTRA to receive the extra keywords in either case, you need only change the _EXTRA to _REF_EXTRA in the definition of the calling routine.

By contrast, the "pass by value" (_EXTRA) keyword inheritance mechanism is useful in the following situations:

1. Your routine needs access to the values of the extra keywords for some reason.
2. You want to ensure that variables specified as keyword parameters are not changed by a called routine.

Example: Keywords Passed by Value

One of the most common uses for the "pass by value" keyword inheritance mechanism is to create "wrapper" routines that extend the functionality of existing routines. In most "wrapper" routines, there is no need to return

set of keywords available to the existing routine in the wrapper routine.

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by John-David T. Smith on Fri, 02 Nov 2001 20:06:55 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
> ----- Original Message -----
> From: "Mark Hadfield" <m.hadfield@niwa.cri.nz>
>>
>> So the routine that's crashing your machine (and mine) is as follows
>> (shortened)...
>>
>> pro test_ref_extra1,_REF_EXTRA=e
>> test_ref_extra2, _REF_EXTRA=e ; Here be dragons!
>> end
>>
>> This isn't merely bad practice, it's wrong. The _REF_EXTRA keyword is only
>> used in routine definitions. When *calling* a routine, the acceptable
> forms
>> are "_EXTRA" and (as of 5.5) "_STRICT_EXTRA".
>>
>> Still, it shouldn't crash IDL. It should be detected as a syntax error.

>>
> Hi Mark
>
> It seemed logical to me that if arguments have to be passed through many
> preprocessing routines to a later routine, that the best way would be by
> reference to avoid copying (as would be the case if the Keyword was
> specified), if the final routine insists on a local copy (by value) why
> should that matter?
> Well I guess I just didnt understand the manual (no change there!). Are you
> saying that this is determined purely by the definition, and that calling
> with _REF_EXTRA as below
>> test_ref_extra2, _REF_EXTRA=e ; Here be dragons! (indeed)
> is supposed to be a syntax error? That was just not clear to me from the
> help (and for that matter the programmer who wrote the syntax parser!!!)
> I am concerned as to which causes passing by reference/value, the definition
> or the call?
> This is almost as horrible as C++ inheritance simple in principle but boy
> can some strange things happen when things get complicated - for now I guess
> will have to stick to _extra which at least is easy to understand :(
>
> roll on coffee time!

As someone unaturally familiar with the intimate details of _REF_EXTRA
(having raised the complaint which led to its creation), let me give a
bit of background in defense of this (admittedly confusing) behavior.

I initially envisioned changing the _EXTRA mechanism to support natively
the default passing behavior of arguments (normal or keyword), which had
been in existence in IDL since time immemorial: everything would be
passed by reference, except indexed arrays, structure field members, and
a few other by-value items known well to anyone more than a casual IDL
user. I reasoned that this default-by-reference behavior was very
familiar to all IDL programmers, and should not provide any technical or
conceptual barriers for extension to *inherited* keyword parameters...
in fact, it was how it should have been implemented to begin with.

The last comment notwithstanding, there was already in place a specific
by-value inherited keyword passing mechanism which provided the
inherited arguments in the form of an _EXTRA structure to intermediate
routines. People started to rely on the specifics of this _EXTRA
structure (yes, this includes me), to do all kinds of interesting and/or
perverse things. The fact that this structure exists as it does, and is
used as it is, meant that no single modification could simultaneously
provide by-reference behavior and access to the old, familiar _EXTRA
structure. So, they settled on an uneasy compromise: an additional
keyword.

However, converting routines to use by-reference keyword inheritance

would then be very awkward: a special-purpose keyword (_REF_EXTRA), would need to be used everywhere. You'd have to keep track of all arbitrary chains of inheritance to make sure you keep the correct "flavor" all along. The RSI developers discovered a trick, however: they could isolate the required change only to the routine definition, and allow all the calling routines (including existing code) to continue to use _EXTRA as-is. And some routines might just do both! Yes, you can pass on a by-reference inherited keyword set to a by-value inheriting routine: the conversion is automatic. (And no, you can't go the other way!).

Had it been done this way from the beginning, we'd have only one mechanism, and it would look and feel like the other argument passing mechanisms in IDL. Such is not the case, and what you see is the compromise which resulted.

So, bottom line rule of thumb: always use _EXTRA in your calls (or at least, never use _REF_EXTRA -- see below). The fact that _REF_EXTRA *could* be used in calls was something of a slip-up, reflecting the later inclusion of the "only-change-in-one-place" feature. The fact that using it crashes IDL remains, of course, a bug.

JD

P.S. With IDLv5.5, yet another _EXTRA keyword appears to muddy the waters: _STRICT_EXTRA. Be assured that it is *very* different in flavor, and as far as I can discern from the single-paragraph "What's New" entry, is meant to be used only in routine *calls*, and never in definitions.

_STRICT_EXTRA simply weeds out extraneous keywords, and signals an error if they occur. This is useful to guard against simple misspellings sneaking in, and not manifesting themselves, e.g.:

```
myroutine,DOMYNICECALCULATION=1

  vs.

myroutine,DOMENICECALCULATION=1
```

neither would raise an error if called through:

```
pro myroutine, _EXTRA=e
  mycalc,_EXTRA=e
end
```

even though presumably mycalc doesn't recognize the second one. You may not get the result you expect, or worse, you may get a result you

erroneously believe, blissfully unaware of your fat fingers. With
_STRICT_EXTRA you can check. Let's just hope it works for both by-value
and by-reference inherited keywords.

Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Mark Hadfield on Sun, 04 Nov 2001 20:54:29 GMT
View Forum Message <> Reply to Message

From: "Martin Downing" <martin.downing@ntlworld.com>
> ...the best way would be byreference to avoid copying. ...Are
> you saying that this is determined purely by the definition,
> and that calling with _REF_EXTRA as below
>> test_ref_extra2, _REF_EXTRA=e ; Here be dragons! (indeed)
> is supposed to be a syntax error?

Yes, that's exactly what I'm saying.

Although it is actually legal (though uncommon) to have regular keywords
beginning with underscores. So I guess strictly speaking it is legal to have
a regular keyword called "_REF_EXTRA". Not a good idea though!

> I am concerned as to which causes passing by reference/value, the
> definition or the call?

The definition!

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--
Posted from clam.niwa.cri.nz [202.36.29.1]
via Mailgate.ORG Server - http://www.Mailgate.ORG
