Subject: Re: Negative indexing and the WHERE function in IDL 8.0
Posted by chris_torrence@NOSPAM on Wed, 18 Aug 2010 16:31:46 GMT
View Forum Message <> Reply to Message

Hi all,

A possible solution is to use the new /NULL keyword to WHERE:

```
data = [1,2,3,4]
data[WHERE(data eq 5, /NULL)] = 999
print, data
      1       2       3       4
```

If /NULL is set, and there are no matches, then WHERE returns the new !NULL "empty array". In IDL 8.0, if you are indexing into an array, and any of the subscripts are !NULL, then the indexing is quietly skipped.

This solution still requires you to examine your old code carefully and retrofit it, but it's a lot easier to add a /NULL to your WHERE calls than to implement the error handling for the -1.

Hope this helps.

-Chris
ITTVIS

Subject: Re: Negative indexing and the WHERE function in IDL 8.0
Posted by svhhaugan on Thu, 19 Aug 2010 12:09:24 GMT
View Forum Message <> Reply to Message

On Aug 18, 6:31 pm, Chris Torrence <gorth...@gmail.com> wrote:
> Hi all,
>
> A possible solution is to use the new /NULL keyword to WHERE:

[..]

> Hope this helps.

Not really. First of all, it isn't IDL 7-compatible. Second, that change by itself only prevents against accidental modification of your data immediately after the WHERE. Program logic will continue past that point based on a false assumption.

How about using "compile_opt strictarrsubs" (which is IDL 7-compatible)
to imply what it always implied in IDL 7 and before: all negative

array subscripts are illegal, whether scalar or arrays.

If only you'd gone with the option that "new semantics go with a new program suffix" (as I recommended when I was asked). As it is now, your customers may find and use stuff from "out there", and it will not work the way it does on IDL 7 when executed by IDL 8. That's wasting a lot of people's time. Take e.g. SolarSoft, almost 50 000 routines that are potentially broken in IDL 8 but not in IDL 7, due to this new feature. And no automatic way to fix it. Inserting a compile_opt statement can easily be done automatically. This will certainly break a few of them, I'm sure, but at least it'll cause a *crash* when something's wrong. Not squash the error message and let the code go on doing something that was never foreseen.

Stein

Subject: Re: Negative indexing and the WHERE function in IDL 8.0
Posted by wlandsman on Thu, 19 Aug 2010 13:29:10 GMT
View Forum Message <> Reply to Message

On Aug 19, 8:09 am, svhhaugan <s.v.h.hau...@gmail.com> wrote:
> On Aug 18, 6:31 pm, Chris Torrence <gorth...@gmail.com> wrote:
>
> How about using "compile_opt strictarrsubs" (which is IDL 7-
> compatible)
> to imply what it always implied in IDL 7 and before: all negative
> array subscripts are illegal, whether scalar or arrays.

That won't work for IDL 8.0 users who want to use the negative scalar indexing, but still need strictarrsubs to give an error when subscripting an array with another array with out of bound indices.

```
IDL> print,!version
{ x86_64 darwin unix Mac OS X 8.0 Jun 17 2010      64      64}

IDL> a = intarr(4)
IDL> a[ [-1,1,2,5] ] = [6,7,8,9]
IDL> print,a
       6       7       8       9
```

--Wayne

P.S. I go back to the very early days (1986) of IDL when WHERE() didn't have the COUNT parameter, and one had to specifically test whether the returned index was equal to -1.

index = where(array)
if index[0] NE -1 then ....

but I had never seen before the use of CATCH to make sure WHERE()
returns a valid value. It seems very convoluted to me, but I suppose
it makes sense if one expects WHERE() to normally return a valid
index, and a single CATCH clause can be used for multiple WHERE()
statements to test for errors. But now those errors won't be
caught...

Subject: Re: Negative indexing and the WHERE function in IDL 8.0
Posted by penteado on Thu, 19 Aug 2010 18:02:14 GMT
View Forum Message <> Reply to Message

On Aug 19, 10:29 am, wlandsman <wlands...@gmail.com> wrote:
> P.S. I go back to the very early days (1986) of IDL when WHERE()
> didn't have the COUNT parameter, and one had to specifically test
> whether the returned index was equal to -1.
>
> index = where(array)
> if index[0] NE -1 then ....

I also used that test a lot, and have seen it often in other people's
programs.

> but I had never seen before the use of CATCH to make sure WHERE()
> returns a valid value. It seems very convoluted to me, but I suppose
> it makes sense if one expects WHERE() to normally return a valid
> index, and a single CATCH clause can be used for multiple WHERE()
> statements to test for errors. But now those errors won't be
> caught...

I only thought about this possibility (when we were discussing the
idea of a new file extension), I never saw it being used. As some say,
it is in principle a choice between having the code ask for permission
(test the result) or ask for forgiveness (catch the error). But I
would guess the catch option is rarely (if at all) used for where()
because it is much easier to test than to make a catch just for that.
A lot of users are not even aware of catch.

Though this is a change in the language that can break code in this
situation, I expect that what will break code much more often is the
introduction of new intrinsic routines (particularly functions), which
is not a language change, and is expected to happen on every new IDL
version. Two examples that have already been mentioned here are the
additions of legend() and list().

In any language, keeping absolute compatibility would cause it to stay nearly frozen, impeding necessary additions and changes. I was also in favor of a new file extension, but in its absence, I find this not to be the end of the world, as long as it is not very often that problems occur. Due to new routines (intrinsic, from installed libraries, and from the own user), the execution environment of an IDL program is not constant across different systems anyway, and it is usually not possible to write a program that will keep working the same, everywhere, and forever (that is, five years in the future).

Subject: Re: Negative indexing and the WHERE function in IDL 8.0
Posted by Greg Hennessy on Fri, 20 Aug 2010 01:35:11 GMT
View Forum Message <> Reply to Message

On 2010-08-19, wlandsman <wlandsman@gmail.com> wrote:
> P.S. I go back to the very early days (1986) of IDL when WHERE()
> didn't have the COUNT parameter, and one had to specifically test
> whether the returned index was equal to -1.

And I go back to having *used* Wayne's code from then. :)

Subject: Re: Negative indexing and the WHERE function in IDL 8.0
Posted by svhhaugan on Fri, 20 Aug 2010 10:36:50 GMT
View Forum Message <> Reply to Message

On Aug 19, 3:29 pm, wlandsman <wlands...@gmail.com> wrote:

That won't work for IDL 8.0 users who want to use the negative
scalar
indexing, but still need strictarrsubs to give an error when
subscripting an array with another array with out of bound indices.

Sure, but in IDL 8.0.1 you can *add* another compile_opt statement
that
ensures you get what you want. You can't do that with IDL 7.

> P.S. I go back to the very early days (1986) of IDL when WHERE()
> didn't have the COUNT parameter, and one had to specifically test
> whether the returned index was equal to -1.
>
> index = where(array)
> if index[0] NE -1 then ....

I go back a bit, too. This is still my preferred way of testing.

> but I had never seen before the use of CATCH to make sure WHERE()
> returns a valid value. It seems very convoluted to me, but I suppose
> it makes sense if one expects WHERE() to normally return a valid
> index, and a single CATCH clause can be used for multiple WHERE()
> statements to test for errors. But now those errors won't be
> caught...

Ok, ok, ok, I regret ever putting that catch statement in
there. People get hung up on it and don't see the forest for
all the trees. It was just to prove a point, that people could be
relying on the documented behavior of negative indices
raising errors, when "things just won't work" with a given
set of parameters.

But I also said:

'the catch,error part would most likely be done "with
human intervention"'.

I.e., just letting the program crash while you're fiddling
with it, giving you a nice and friendly command line inside
an emacs window where you can find out what the heck went wrong,
where, and why.

It is, after all, an "Interactive" data language. Not
everyone using IDL are *required* to write program suites
that work for all possible parameters and situations, so it
can be sold to a paying customer. If that was a requirement
then IDL would be entirely unsuitable for serious research,
IMO.

Cheers,
Stein
