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).