Subject: Re: Functions and arrays
Posted by thompson on Wed, 04 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

rigby@crd.ge.com (Wayne Rigby) writes:

> In article 6ll@post.gsfc.nasa.gov, thompson@orpheus.nascom.nasa.gov (William Thompson) writes:
>>
>> I've just run into a serious limitation in the way IDL handles functions and
>> arrays. I've always been aware that there was an ambiguity in the IDL syntax,
>> ...
>> It is my opinion that IDL should be tightened in its handling of arrays and
>> functions. If a procedure contains a variable with a given name, then it
>> should be unambiguous that one is referencing that variable, and not some
>> function which happens to share the same name.
>> ...
>> William Thompson

> Please complain directly to RSI about this! We ran into this about a year
> ago, and were unable to get past the first level support folks telling us that
> this was an inevitable consequence of IDL's weak typing.

I sent a copy of the same message to RSI's support people, and they promised to get back to me on it. I'm sure that the messages posted here will have some effect on their response.

Bill Thompson

Subject: Re: Functions and arrays
Posted by steinhh on Wed, 04 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

> Please complain directly to RSI about this! We ran into this about a year
> ago, and were unable to get past the first level support folks telling us that
> this was an inevitable consequence of IDL's weak typing.

People at RSI (and not just support people) do read this newsgroup (and act on it as well: XPAD/YPAD/SPACE=0 will once again rule, in IDL v 5.0!), so this place may very well be the best place for a campaign to remove this quite serious (and dangerous) "feature". I do think that we need to show some enthusiasm to make it happen, though (i.e., the more people agreeing about this being a rather nasty thing that they'd like to see removed, the better).

I just cannot see that this is actually an *inevitable* consequence of IDL's typing... it could be an inevitable consequence of a single-pass/single-

statement compiler, though. That, however, has nothing to do with the language as such.

Now, assume I want to refer to an element of an array called "test1" somewhere inside a program, e.g.,:

```
value = test1(0)
```

Assuming, of course, that I do *not* write a program that references an undefined variable on purpose, is there *any* way of making this line a non-crashing statement *without* referring to the name "test1" (note: no parentheses) *anywhere* inside the same procedure?

Unless someone can actually show a compelling example of this, I'd say that there is no problem in altering the compiler to behave reasonably, e.g., to automatically detect which names are actually variables, and act accordingly.

Due to the fact that one may write non-crashing programs referencing array elements in statements placed *before* any assignment statement, it's necessary to have either a two-pass compiler or to have some back-tracking for this to work (this must already be done for e.g., the labels in GOTO statements).

If this is too much of a problem, could we at least have a "forward_array" or "forward_variable" statement ? :-)

Or at least get a compile time error about the possible mixup, something like "Error: test1 interpreted as a function in line 5, but as a variable in line 10".

Stein Vidar H. Haugan

Subject: Re: Functions and arrays
Posted by gunter on Wed, 04 Dec 1996 08:00:00 GMT

View Forum Message <> Reply to Message

William Thompson (thompson@orpheus.nascom.nasa.gov) wrote:
[see previous post for details]

I find this behaviour to be irresponsible on the part of IDL. To really stress the point, do the following. (There is no new info just a better illustration of the problem for those new to IDL)

Insert a STOP command in the function test1, after assigning the values to the test2 array:

```
:  function test1, i
:  test2 = [1,3,7]
 STOP
:  a = test2(i)
:  return, a
:  end
```

Then compile test2 first, then test1 and run test1 via
  print, test1(0)

When you get back the command prompt, type HELP and you'll see that there is an array named test2 and lower you will see that there is a function named test2.

```
 (idl)% print, test1(0)
 % Stop encountered:  TEST1               3 /TEST1.pro
 (idl)% help
 % At  TEST1               3 /TEST1.pro
 %     $MAIN$
 Code area used: 100% (200/200),  Symbol area used: 21% (40/184)
 # local variables: 3,  # parameters: 1
 A              UNDEFINED = <Undefined>
 TEST2          INT       = Array(3)
 I              INT       =        0
 Compiled Procedures:
    $MAIN$  LOADCT
 Compiled Functions:
    TEST1    TEST2
```

Now, type
  print, test2
and you'll see the values in the array. If you then type
  print, test2(0)
You will get the value of the function test2 (-1 in this case).

Again, there's nothing new that previous posts haven't told us, but I find the behaviour rather silly.
--
david gunter
http://www.mcs.anl.gov/people/gunter/

"When you are a Bear of Very Little Brain, and you Think of Things, you find sometimes that a Thing which seemed very Thingish inside you is quite different when it gets out into the open and has other people looking at it."
- A.A. Milne, "The House At Pooh Corner"

Subject: Re: Functions and arrays
Posted by rigby on Wed, 04 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article 6ll@post.gsfc.nasa.gov, thompson@orpheus.nascom.nasa.gov (William Thompson) writes:
>
> I've just run into a serious limitation in the way IDL handles functions and
> arrays. I've always been aware that there was an ambiguity in the IDL syntax,
> ...
> It is my opinion that IDL should be tightened in its handling of arrays and
> functions. If a procedure contains a variable with a given name, then it
> should be unambiguous that one is referencing that variable, and not some
> function which happens to share the same name.
> ...
> William Thompson

Please complain directly to RSI about this! We ran into this about a year ago, and were unable to get past the first level support folks telling us that this was an inevitable consequence of IDL's weak typing.

We even gave them an example of how their own routines could be broken. The routine poly.pro in $IDL_DIR/lib happens to use the name C for one of its formal arguments:

```
        FUNCTION POLY,X,C
```

If a user should compile a function C before POLY is compiled, he/she will get the wrong answer from POLY. Silently.

```
IDL> .run
- function c, arg
- return, arg-1000
- end
 Compiled module: C.
IDL> print, poly(1,[1,1])
 Compiled module: POLY.
      -1999              (should be 2).
```

There's no reasonable way to defend yourself against this either.
Are you supposed to examine the formal argument list of every routine you use??

Wayne Rigby
GE Corporate Research and Development
rigby@crd.ge.com

Subject: Re: Functions and arrays
Posted by greenwoodde on Thu, 05 Dec 1996 08:00:00 GMT

In a previous article, William Clodius <wclodius@lanl.gov> wrote:

> Allowing '[]' to replace '()' may be a good idea, but it can provide a
> maintenance problem for old code.

Please don't underestimate the problem of maintaining old code. We have some recent experience with this. Back in the dark ages - before IDL even had version numbers - we wrote an external library. Then IDL 2 came along and we didn't have the time to update our library. With the help of RSI we've now done that, but the result is that we effectively went from IDL "v1" to v4 in one jump. Because several non-compatible changes were introduced in the meantime we've had a number of unpleasant surprises when people tried to dust off old code. In fact, I have to maintain a copy of "v1" just so we can run that old code :-(

Now, personally I like the idea of replacing "()" with "[]" for arrays. My suggestion would be to introduce "[]" as an *alternate* way of specifying arrays. In fact, it could be documented as the *preferred* way - to avoid the issues that started this thread.

Dave

Dave Greenwood Internet: Greenwoodde@ORNL.GOV
Oak Ridge National Lab %STD-W-DISCLAIMER, I only speak for myself

---

Subject: Re: Functions and arrays
Posted by William Clodius on Thu, 05 Dec 1996 08:00:00 GMT

Wayne Rigby wrote:
>
> <snip>
>
> Please complain directly to RSI about this! We ran into this about a year
> ago, and were unable to get past the first level support folks telling us that
> this was an inevitable consequence of IDL's weak typing.
>
> <snip>

The effect described is not due to weak typing, it is due to poor scoping. IDL should only be attempting to interpret a name as an external function reference if the name has not been previously defined in the scope of the procedure. Providing the name as an argument or having the name initially appeare on the left hand side of an assignment should subsequently define it within the scope. It appears that either

they have not clearly differentiated the two name spaces, or they are improperly searching the external name space first. Note that having two name spaces may have a performance impact on compilation as function calls would require two searches rather than one.

My suspicion is that they have not set up two separate name spaces under the assumption that name conflicts would be extremely rare. Fixing the scoping issue may require an extensive rewrite of their system.

Allowing '[]' to replace '()' may be a good idea, but it can provide a maintenance problem for old code.

--

William B. Clodius Phone: (505)-665-9370
Los Alamos Nat. Lab., NIS-2 FAX: (505)-667-3815
PO Box 1663, MS-C323 Group office: (505)-667-5776
Los Alamos, NM 87545 Email: wclodius@lanl.gov

NIS-2: Nonproliferation & International Security Div./
Astrophysics & Radiation Measurements Group

Subject: Re: Functions and arrays
Posted by thompson on Thu, 05 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Peter Mason <peterm@demsyd.syd.dem.csiro.au> writes:

> On 4 Dec 1996, Stein Vidar Hagfors Haugan wrote:
>> People at RSI (and not just support people) do read this newsgroup (and
>> act on it as well: XPAD/YPAD/SPACE=0 will once again rule, in IDL v 5.0!),
>> so this place may very well be the best place for a campaign to remove
>> this quite serious (and dangerous) "feature". I do think that we need to
>> show some enthusiasm to make it happen, though (i.e., the more people
>> agreeing about this being a rather nasty thing that they'd like to see
>> removed, the better).

> I'd also really like to see this ambiguity sorted out.

>> Or at least get a compile time error about the possible mixup,
>> something like "Error: test1 interpreted as a function in line 5,
>> but as a variable in line 10".

> I think that this would be a good idea. It wouldn't bust any code that
> wasn't already hovering on the edges of "busthood", and it would catch many
> of the ambiguities. When requested to compile a function, IDL could stop with

> an error if a variable of the same name already existed. ...

There seems to be a assumption here that the *PROGRAMMER* is forming an ambiguity by trying to use the same name for both a variable and a function in the same routine. I argue that it's *IDL* which is responsible for the ambiguity. The situation I ran into was when the software was written in a self-consistent manner--the name was intended to refer to a variable throughout the routine. However, IDL on its own decided to sometimes interpret the call as a variable (correct) and sometimes as a function (incorrect), depending on what applications were started first in the IDL session.

The correction to this is quite simple. If a name is used for a variable in a subroutine, then it refers to that variable throughout that subroutine. The fact that it may be used to refer to a function in some other routine is completely immaterial. I'm sure that the IDL syntax parser could be made to be unambiguous here--as somebody pointed out it just requires a double pass through the source code.

The idea that a correctly written and working piece of code could be broken by adding a function in a completely unrelated piece of code is totally repugnant to me.

Bill Thompson

Subject: Re: Functions and arrays
Posted by Mirko Vukovic on Thu, 05 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Peter Mason wrote:
stuff deleted ..
> But I think that a proper cure would involve a basic syntax change, like
> using [] instead of () on arrays.
more stuff deleted
> Peter Mason

How about
a=foo,blah,blah,blah

I often do make a mistake of writing that, only to be reminded by IDL of a syntax error. And just in case foo is a procedure, well, return 1 to a.
--
Mirko Vukovic, Ph.D 3075 Hansen Way M/S K-109
Varian Associates Palo Alto, CA, 94304
415/424-4969 mirko.vukovic@varian.grc.com

Subject: Re: Functions and arrays
Posted by brian.jackel on Thu, 05 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article <Pine.SUN.3.91.961205155301.9297C-100000@demsyd.syd.dem.csiro.au> Peter Mason <peterm@demsyd.syd.dem.csiro.au> writes:

> But I think that a proper cure would involve a basic syntax change, like
> using [] instead of () on arrays. If RSI proposed this, I think I would
> actually consider supporting it. (I'm sure I'm pretty much alone in this
> opinion - just thought I'd mention it.)

Actually, I also think it's a good idea. More work in the short term, less hassle in the long term.

Brian Jackel

Subject: Re: Functions and arrays
Posted by Anonymous on Thu, 05 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Originally posted by: XXX (Steve)

In article <32a695b9.0@epflnews.epfl.ch>, llobet@crpp.uhd.epfl.ch wrote:

> In article <Pine.SUN.3.91.961205155301.9297C-100000@demsyd.syd.dem.csiro.au>,
> Peter Mason <peterm@demsyd.syd.dem.csiro.au> writes:
>
> =But I think that a proper cure would involve a basic syntax change, like
> =using [] instead of () on arrays. If RSI proposed this, I think I would
> =actually consider supporting it. (I'm sure I'm pretty much alone in this
> =opinion - just thought I'd mention it.)
>
I think this is a good idea. I have no idea how hard it would be to implement or what it would do to backwards compatibility. But by itself it seems like a good idea.

Subject: Re: Functions and arrays
Posted by llobet on Thu, 05 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article <Pine.SUN.3.91.961205155301.9297C-100000@demsyd.syd.dem.csiro.au>, Peter Mason <peterm@demsyd.syd.dem.csiro.au> writes:

=But I think that a proper cure would involve a basic syntax change, like
=using [] instead of () on arrays. If RSI proposed this, I think I would

=actually consider supporting it. (I'm sure I'm pretty much alone in this
=opinion - just thought I'd mention it.)

I would second this.

And yes, I am aware of the main implication: need to review all the existing
software. But it should be possible to automate it in most of the cases.

-xavier

Subject: Re: Functions and arrays
Posted by Peter Mason on Thu, 05 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

On 4 Dec 1996, Stein Vidar Hagfors Haugan wrote:
> People at RSI (and not just support people) do read this newsgroup (and
> act on it as well: XPAD/YPAD/SPACE=0 will once again rule, in IDL v 5.0!),
> so this place may very well be the best place for a campaign to remove
> this quite serious (and dangerous) "feature". I do think that we need to
> show some enthusiasm to make it happen, though (i.e., the more people
> agreeing about this being a rather nasty thing that they'd like to see
> removed, the better).

I'd also really like to see this ambiguity sorted out.

> Or at least get a compile time error about the possible mixup,
> something like "Error: test1 interpreted as a function in line 5,
> but as a variable in line 10".

I think that this would be a good idea. It wouldn't bust any code that
wasn't already hovering on the edges of "busthood", and it would catch many
of the ambiguities. When requested to compile a function, IDL could stop with
an error if a variable of the same name already existed. When requested to
create a new variable, IDL could stop with an error if a function of the same
name had already been compiled, or had been declared in a FORWARD_FUNCTION
statement.
This would still have some holes. Functions are often coded in their own
source files (one function per file) to facilitate automatic compilation.
One would have to force the compilation of such functions with RESOLVE (or
such), or flag them with FORWARD_FUNCTION, to be sure of things.

But I think that a proper cure would involve a basic syntax change, like
using [] instead of () on arrays. If RSI proposed this, I think I would
actually consider supporting it. (I'm sure I'm pretty much alone in this
opinion - just thought I'd mention it.)

Peter Mason

Subject: Re: Functions and arrays
Posted by Tim Patterson on Fri, 06 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

William Clodius wrote:
>

>
> Allowing '[]' to replace '()' may be a good idea, but it can provide a
> maintenance problem for old code.
>

IMHO such a change would cause too many problems with maintenance.
I'm working on a very large software package (10's of thousands
of lines of IDL code) that has to run on both highest
and lowest common demoninator IDL systems (currently 4.0.1
is the lcd). If IDL 5.0+ changed the syntax, I'd have to have
2 copies of the code, not to mention the time it would take
to update the old code to the new standard. Unfortunately, due
to IDL's high prices at present, I couldn't persuade all my
users to upgrade to IDL 5.0.

I think having a choice of either () or [] would also be
confusing and bad practise. It seems to me that the compiler
checking for and flagging/resolving these problems is the
best solution. Especially as no programmer would try and give an
array and a function the same name in the same routine, would they? :)

 Tim

Subject: Re: Functions and arrays
Posted by haimov on Fri, 06 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article <584821$mv6@ratatosk.uio.no>, steinhh@amon.uio.no says...

> People at RSI (and not just support people) do read this newsgroup (and
> act on it as well: XPAD/YPAD/SPACE=0 will once again rule, in IDL v 5.0!),
> so this place may very well be the best place for a campaign to remove
> this quite serious (and dangerous) "feature". I do think that we need to
> show some enthusiasm to make it happen, though (i.e., the more people

> agreeing about this being a rather nasty thing that they'd like to see
> removed, the better).

I agree completely that this is a bad "feature". I have had several occasions when this caused me a significant waste of time.

On the other hand I never had such a problem in MATLAB. Their compiler
uses a precedence to uniquely define occurences of same name functions and variables.
Variables always have precedence. This way I don't need to be very careful when choosing
variable names for as long as I won't need the corresponding functions.
However even with this convention some confusion is possible. So it would be nice if
the compiler can issue warnings of coexisting same name functions/procedures and variables.

Samuel Haimov, Ph.D.
University of Wyoming

Subject: Re: Functions and arrays
Posted by steinhh on Fri, 06 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article <5877p7$6gp@post.gsfc.nasa.gov>, thompson@orpheus.nascom.nasa.gov (William Thompson) writes:
|>
|>On 4 Dec 1996, Stein Vidar Hagfors Haugan wrote:
|> >> Or at least get a compile time error about the possible mixup,
|> >> something like "Error: test1 interpreted as a function in line 5,
|> >> but as a variable in line 10".
|>
|> Peter Mason <peterm@demsyd.syd.dem.csiro.au> writes:
|> >I think that this would be a good idea. It wouldn't bust any code that
|> >wasn't already hovering on the edges of "busthood", and it would catch many
|> >of the ambiguities. When requested to compile a function, IDL could stop with
|> >an error if a variable of the same name already existed. ...
|>
|> There seems to be a assumption here that the *PROGRAMMER* is forming an
|> ambiguity by trying to use the same name for both a variable and a function in
|> the same routine. I argue that it's *IDL* which is responsible for the
|> ambiguity. The situation I ran into was when the software was written in a
|> self-consistent manner--the name was intended to refer to a variable throughout
|> the routine. However, IDL on its own decided to sometimes interpret the call
|> as a variable (correct) and sometimes as a function (incorrect), depending on
|> what applications were started first in the IDL session.

Just to clear up any misunderstandings: I do not see this in any way as a
programmer error. It is a *compiler* error. My preferred solution was exactly the
same as you state, Bill:

|> The correction to this is quite simple. If a name is used for a variable in a
|> subroutine, then it refers to that variable throughout that subroutine. The

The reason I included other options for minimizing (though not eliminating) the problem is that a rewrite of IDL's compiler procedures might be too complicated to appear any time soon (version 6? or 7?). Consider the following:

```
pro compiler_trouble

  while n_elements(value) eq 0 do begin
    if n_elements(dummy) eq 0 then begin
       value = arr(0)
    end

    dummy = 1
    arr = fltarr(5)
  endwhile
end
```

and couple that with the possible existence of a function "arr".

Although the definition (or lack thereof) of IDL as a language does not create any problems in resolving the conflict in the only reasonable fashion (always interpreting arr as a variable inside pro compiler_trouble), I fear that the current IDL compiler is not up to that task without at least some medium-sized rewrite.

So, in case the only decent thing cannot be done soon, I'd like to see *something* done about the problem, as a temporary solution.

Including both a "forward_variable" statement *and* a compiler message about a potential *compiler* error would at least cut the time spent searching for mysterious bugs down to almost zero. Granted, it would still allow a lot of perfectly fine routines to suddenly become non-functional simply because someone happened to add some function, and I agree, that's not acceptable. But a temporary solution is better than *no* solution.

A "forward_variable" statement could of course be implemented with no additions to the language, with a statement like "if 0 then arr = 0". That shouldn't leave much room for doubt if the compiler would listen.....

Stein Vidar

Subject: Re: Functions and arrays
Posted by David R. Klassen on Sat, 07 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

William Clodius wrote:
> scoping. IDL should only be attempting to interpret a name as an
> external function reference if the name has not been previously defined
> in the scope of the procedure. Providing the name as an argument or
See, right now my problem is just the opposite. I reference a function
that IDL is trying to interpret as an array even though the "array" has
not been previously referenced. I can't seem to understand why the
program
can't retrieve the function...

--
David R. Klassen
Department of Physics and Astronomy
University of Wyoming
Box 3905 University Station
Laramie WY 82071
http://faraday.uwyo.edu/grads/dklassen/
drk@uwyo.edu