Subject: reverse problem
Posted by news.qwest.net on Fri, 04 Nov 2005 00:24:15 GMT
View Forum Message <> Reply to Message

Hi everybody,
I've gone insane. Here is the evidence:

IDL> print,reverse([1,2,3,4,5])

1 1 1 1 1

Why the heck is reverse giving me an array of ones, instead
of reversing the array?
I think it used to work ok, since I came across this problem
in an old color table procedure (where I can select to reverse the
color table).

Cheers,
bob

Subject: Re: reverse problem
Posted by news.qwest.net on Fri, 04 Nov 2005 15:38:45 GMT
View Forum Message <> Reply to Message

I think I have resolved the issue. I'd guess it is a namespace problem.
I had a keyword "reverse" as well as the function "reverse()".
This is using the default compiler options (i.e. I am not setting anything).

So if you start of a fresh IDL, and type
IDL> .run namespacecrash

% Compiled module: NAMESPACECRASH.

% Compiled module: $MAIN$.

0.000000 1.00000 2.00000 3.00000 4.00000

1 1 1 1 1

Here is the routine:

function namespacecrash,array,reverse = reverse

```
if keyword_set(reverse) then begin
 array = reverse(array)
endif

return,array
end

array = findgen(5)

print,namespacecrash(array)

print,namespacecrash(array,/rev)
end
```

My results are:

```
% Compiled module: NAMESPACECRASH.

% Compiled module: $MAIN$.

IDL> .GO

0.000000 1.00000 2.00000 3.00000 4.00000

1 1 1 1 1

IDL> help,!version,/st

** Structure !VERSION, 8 tags, length=76, data length=76:

ARCH STRING 'x86'

OS STRING 'Win32'

OS_FAMILY STRING 'Windows'

OS_NAME STRING 'Microsoft Windows'

RELEASE STRING '6.1.1'

BUILD_DATE STRING 'Oct 11 2004'

MEMORY_BITS INT 32

FILE_OFFSET_BITS
```

Subject: Re: reverse problem
Posted by liamgumley on Fri, 04 Nov 2005 15:41:27 GMT
View Forum Message <> Reply to Message

Bob,

Is it possible that you have another file named reverse.pro somewhere
in your IDL path? Here's one way you can find which version of
reverse.pro is being compiled:

IDL> doc_library, 'reverse'
----- Documentation for /Applications/rsi/idl_6.1/lib/reverse.pro -----

Cheers,
Liam.
Practical IDL Programming
http://www.gumley.com/

Subject: Re: reverse problem
Posted by Paolo Grigis on Fri, 04 Nov 2005 16:25:17 GMT
View Forum Message <> Reply to Message

R.G. Stockwell wrote:
> I think I have resolved the issue. I'd guess it is a namespace problem.
> I had a keyword "reverse" as well as the function "reverse()".
> This is using the default compiler options (i.e. I am not setting anything).

Another example which shows how bad it is to have round
parenthesis syntax allowed for both arrays and functions...

one would guess that probably the keyboard of whomever
invented the first version of IDL syntactic rules lacked
the keys for [] ;-)

Ciao,
Paolo

P.S.
print,namespacecrash(array,rev=2) is also quite funny

>
>
> So if you start of a fresh IDL, and type

> IDL> .run namespacecrash
>
> % Compiled module: NAMESPACECRASH.
>
> % Compiled module: $MAIN$.
>
> 0.000000 1.00000 2.00000 3.00000 4.00000
>
> 1 1 1 1 1
>
>
>
>
> Here is the routine:
>
> function namespacecrash,array,reverse = reverse
>
> if keyword_set(reverse) then begin
> array = reverse(array)
> endif
>
> return,array
> end
>
> array = findgen(5)
>
> print,namespacecrash(array)
>
> print,namespacecrash(array,/rev)
> end
>
>
>
> My results are:
> % Compiled module: NAMESPACECRASH.
>
> % Compiled module: $MAIN$.
>
> IDL> .GO
>
> 0.000000 1.00000 2.00000 3.00000 4.00000
>
> 1 1 1 1 1
>
> IDL> help,!version,/st
>
> ** Structure !VERSION, 8 tags, length=76, data length=76:
>

> ARCH STRING 'x86'
>
> OS STRING 'Win32'
>
> OS_FAMILY STRING 'Windows'
>
> OS_NAME STRING 'Microsoft Windows'
>
> RELEASE STRING '6.1.1'
>
> BUILD_DATE STRING 'Oct 11 2004'
>
> MEMORY_BITS INT 32
>
> FILE_OFFSET_BITS
>
> INT 64
>
>

Subject: Re: reverse problem
Posted by MarioIncandenza on Fri, 04 Nov 2005 16:59:06 GMT
View Forum Message <> Reply to Message

In case anyone is unaware, IDL now has a line you can put into any
procedure
COMPILE_OPT IDL2;
which will strictly enforce the difference between [] and ().
Incidentally, there is another line,
COMPILE_OPT LOGICAL_PREDICATE;
which causes boolean tests to be evaluated as they are in Perl, which I
find more intuitive than IDL's rules. For details, see the manpage
under COMPILE_OPT.

Subject: Re: reverse problem
Posted by news.qwest.net on Fri, 04 Nov 2005 17:04:00 GMT
View Forum Message <> Reply to Message

<liamgumley@gmail.com> wrote in message
news:1131118887.329659.247860@g43g2000cwa.googlegroups.com.. .
> Bob,
>
> Is it possible that you have another file named reverse.pro somewhere
> in your IDL path? Here's one way you can find which version of
> reverse.pro is being compiled:

>
> IDL> doc_library, 'reverse'
> ----- Documentation for /Applications/rsi/idl_6.1/lib/reverse.pro -----
>
> Cheers,
> Liam.
> Practical IDL Programming
> http://www.gumley.com/

Good tip Liam. The only "reverse" function is the RSI one.

As Paolo points out, the routine is returning the keyword value,
replicated to the size of the input array. Neat!
So, it is treating reverse as a single element array, then accessing
it with the input array as an index (which exceeds the "reverse" array
size, and by IDL's default rule, it replicates the last number).

example:
IDL> x = 1

IDL> print, x(findgen(10))

1 1 1 1 1 1 1 1 1 1

But the really really strange thing is that, as I showed in my original
post,
that the problem was persistent. After running the routine, the keyword
"reverse"
escaped the local scope of the routine, and came to main level (and thus
overwriting the
reverse function at the main level). Of course, this is how keywords are
supposed to work,
but it was kind of a surprise.

Cheers,
bob

Subject: Re: reverse problem
Posted by David Fanning on Fri, 04 Nov 2005 17:22:02 GMT
View Forum Message <> Reply to Message

R.G. Stockwell writes:

> As Paolo points out, the routine is returning the keyword value,

> replicated to the size of the input array. Neat!
> So, it is treating reverse as a single element array, then accessing
> it with the input array as an index (which exceeds the "reverse" array
> size, and by IDL's default rule, it replicates the last number).
>
> example:
> IDL> x = 1
>
> IDL> print, x(findgen(10))
>
> 1 1 1 1 1 1 1 1 1 1
>
>
> But the really really strange thing is that, as I showed in my original
> post,
> that the problem was persistent. After running the routine, the keyword
> "reverse"
> escaped the local scope of the routine, and came to main level (and thus
> overwriting the
> reverse function at the main level). Of course, this is how keywords are
> supposed to work,
> but it was kind of a surprise.

Oh, this is neat! This is now my favorite example
in my Pass-by_Reference, Pass-by-Value lecture. :-)

Cheers,

David

P.S. By the way, putting Compile_Opt idl2 in your IDL startup
file goes a LONG way to sorting out a lot of these kinds of
problem.
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: reverse problem
Posted by news.qwest.net on Fri, 04 Nov 2005 18:42:40 GMT
View Forum Message <> Reply to Message

"David Fanning" <david@dfanning.com> wrote in message
news:MPG.1dd546a9ef67732f989693@news.frii.com...
...
> P.S. By the way, putting Compile_Opt idl2 in your IDL startup
> file goes a LONG way to sorting out a lot of these kinds of

> problem.
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

Yeah, I've always wanted to change that. However, it goes against my geological sedimentation style of programming. I still have every program I have ever written (and a quick look shows that I have 2278 files in my personal IDL library). These have carefully been layered over the years, and some of the older one are starting to fossilize. The point being, many routine will break because I have all kinds of parenthesis function calls, and
other things that surely are no longer allowed.

One of these days I will have to extract my useful functions from that library,
if only I were less lazy ^H^H^H^H busy :)

Cheers,
bob

PS maybe my code is not fossilizing, and perhaps an Intelligent Designer is placing the code there. However, I see no evidence of intelligence in those
files, they appear to have been randomly generated and debugged into existence.

Subject: Re: reverse problem
Posted by David Fanning on Fri, 04 Nov 2005 20:27:10 GMT
View Forum Message <> Reply to Message

R.G. Stockwell writes:

> Yeah, I've always wanted to change that. However, it goes against my
> geological sedimentation style of programming. I still have every program
> I have ever written (and a quick look shows that I have 2278 files in my
> personal IDL library). These have carefully been layered over the years,
> and some of the older one are starting to fossilize. The point being, many
> routine will break because I have all kinds of parenthesis function calls,
> and
> other things that surely are no longer allowed.

The great thing (I guess) about the compiler options
is that they only take effect in the program module

in which they are declared. Thus, if you declare
a compile_opt at the main IDL level, they are not
in effect for the procedures and functions you run
at the main level, unless you have placed a
compile_opt command in those procedures and functions.

This makes is possible to move forward with new ideas
(that is, put a COMPILE_OPT statement in all the *new*
procedures and functions you write) and still be able to
run your rapidly fossilizing ones. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: reverse problem
Posted by news.qwest.net on Fri, 04 Nov 2005 21:16:25 GMT
View Forum Message <> Reply to Message

"David Fanning" <david@dfanning.com> wrote in message
news:MPG.1dd571fda9986354989694@news.frii.com...
> The great thing (I guess) about the compiler options
> is that they only take effect in the program module
> in which they are declared. Thus, if you declare
> a compile_opt at the main IDL level, they are not
> in effect for the procedures and functions you run
> at the main level, unless you have placed a
> compile_opt command in those procedures and functions.
>
> This makes is possible to move forward with new ideas
> (that is, put a COMPILE_OPT statement in all the *new*
> procedures and functions you write) and still be able to
> run your rapidly fossilizing ones. :-)
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

Wow, I did not know that. That is great (I guess).
I had thought the compiler option would have been a
global option, affecting all scopes.

hmmmm, but now that you mention it, I think I'd rather
have it as a global option and change () to [] when it breaks,
than put a compiler option in every piece of code I write.

I've known that they exist, but I'll have to seriously look into these
compiler options.
Perhaps a lil of the RTFM is in order.

Cheers,
bob

Subject: Re: reverse problem
Posted by savoie on Fri, 04 Nov 2005 22:03:43 GMT
View Forum Message <> Reply to Message

"R.G. Stockwell" <no@email.please> writes:

> hmmmm, but now that you mention it, I think I'd rather
> have it as a global option and change () to [] when it breaks,
> than put a compiler option in every piece of code I write.

I'll second this. Additionally, it's against pleasing aesthetics to have to
put a complier option at the beginning of every single freaking routine.

Matt

--
Matthew Savoie - Scientific Programmer
National Snow and Ice Data Center
(303) 735-0785 http://nsidc.org

Subject: Re: reverse problem
Posted by David Fanning on Fri, 04 Nov 2005 22:31:02 GMT
View Forum Message <> Reply to Message

Matt Savoie writes:

> I'll second this. Additionally, it's against pleasing aesthetics to have to

> put a complier option at the beginning of every single freaking routine.

I think the thought was it is better for your blood
pressure to do it this way than to have random programs
breaking at just the wrong time. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
