Subject: Re: Invalid indices?
Posted by David Fanning on Tue, 27 Oct 2009 18:00:48 GMT
View Forum Message <> Reply to Message

Y.T. writes:

> So I'm kinda living under a rock:
> IDL Version 6.3, Microsoft Windows (Win32 x86 m32)
>
> so I'm curious whether this is intended/expected behaviour or a bug or
> what (and whether it has change in recent years):
>
> IDL> t = lindgen(7)
> IDL> print,t
> 0 1 2 3 4
> 5 6
>
> IDL> n=5*indgen(5)
> IDL> print,n
> 0 5 10 15 20
>
> IDL> t[n] = 100
> IDL> print,t
> 100 1 2 3 4
> 100 100
>
> So element number 0 got set to 100 (OK), element number 5 got set to
> 100 and ... element number 6 also got set to 100?
>
> Why is that? I understand that I'm specifying elements "out of
> range" (number 10 and 15 etc) - is that the reason? Is this
> documented? It took me by surprise...

The explanation (I believe!) can be found in this article:

http://www.dfanning.com/code_tips/lhsvsrhs.html

The situation is known. And, yes, it surprises a LOT of people. :-)

Cheers,

David
--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Invalid indices?
Posted by Foldy Lajos on Tue, 27 Oct 2009 18:02:39 GMT
View Forum Message <> Reply to Message

On Tue, 27 Oct 2009, Y.T. wrote:

> So I'm kinda living under a rock:
> IDL Version 6.3, Microsoft Windows (Win32 x86 m32)
>
> so I'm curious whether this is intended/expected behaviour or a bug or
> what (and whether it has change in recent years):
>
> IDL> t = lindgen(7)
> IDL> print,t
> 0 1 2 3 4
> 5 6
>
> IDL> n=5*indgen(5)
> IDL> print,n
> 0 5 10 15 20
>
> IDL> t[n] = 100
> IDL> print,t
> 100 1 2 3 4
> 100 100
>
> So element number 0 got set to 100 (OK), element number 5 got set to
> 100 and ... element number 6 also got set to 100?
>
> Why is that? I understand that I'm specifying elements "out of
> range" (number 10 and 15 etc) - is that the reason? Is this
> documented? It took me by surprise...
>

There is no out of range error for array subscripts, they are always clipped. From the docs:

Elements of the subscript array that are negative or larger than the highest subscript are clipped to the target array boundaries. Note that a common error is to use a negative scalar subscript (e.g., A[-1]). Using this type of subscript causes an error. Negative array subscripts (e.g., A[[-1]]) do not cause errors.

regards,
lajos

Subject: Re: Invalid indices?

Posted by David Fanning on Tue, 27 Oct 2009 18:27:00 GMT
View Forum Message <> Reply to Message

Lajos writes:

> There is no out of range error for array subscripts, they are always
> clipped. From the docs:
>
> Elements of the subscript array that are negative or larger than the
> highest subscript are clipped to the target array boundaries. Note that a
> common error is to use a negative scalar subscript (e.g., A[-1]). Using
> this type of subscript causes an error. Negative array subscripts (e.g.,
> A[[-1]]) do not cause errors.

Thanks. Much easier than reading the article. I tried it
about a half hour ago, and my eyes are still crossed. :-(

Cheers,

David
--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Invalid indices?
Posted by Michael Galloy on Tue, 27 Oct 2009 18:48:26 GMT
View Forum Message <> Reply to Message

Y.T. wrote:
> So I'm kinda living under a rock:
> IDL Version 6.3, Microsoft Windows (Win32 x86 m32)
>
> so I'm curious whether this is intended/expected behaviour or a bug or
> what (and whether it has change in recent years):
>
> IDL> t = lindgen(7)
> IDL> print,t
> 0 1 2 3 4
> 5 6
>
> IDL> n=5*indgen(5)
> IDL> print,n
> 0 5 10 15 20
>
> IDL> t[n] = 100
> IDL> print,t

> 100 1 2 3 4
> 100 100
>
> So element number 0 got set to 100 (OK), element number 5 got set to
> 100 and ... element number 6 also got set to 100?
>
> Why is that? I understand that I'm specifying elements "out of
> range" (number 10 and 15 etc) - is that the reason? Is this
> documented? It took me by surprise...

If you would rather have an error thrown in this case, do

```
compile_opt strictarrsubs
```

before you do your indexing:

```
IDL> d = findgen(10)
IDL> print, d[[-1, 0, 5, 10, 11]]
      0.00000      0.00000      5.00000      9.00000      9.00000
IDL> compile_opt strictarrsubs
IDL> print, d[[-1, 0, 5, 10, 11]]
% Array used to subscript array contains out of range subscript: D.
% Execution halted at: $MAIN$
```

Mike
--
www.michaelgalloy.com
Research Mathematician
Tech-X Corporation

Subject: Re: Invalid indices?
Posted by Brian Larsen on Wed, 28 Oct 2009 23:02:29 GMT
View Forum Message <> Reply to Message

There are quite a few compile options out there, some I use normally
and some I don't. THis second I am worrying about execution speed
differences for compile options.

I am tempted to use strictarrsubs and strictarr in all my codes,
anyone have any thoughts on if there is a speed difference with any
compile options? I will do a test also but I'm curious of others
thoughts and experiences.

Using this simple code I see:

```
PRO run_test
a1 = findgen(1000)
```

```
a2 = findgen(1000)
t0 = systime(/sec)
  FOR i = 0UL, 1000 DO BEGIN
    s3 = a1#a2
  ENDFOR
print, systime(/sec)-t0
END
```

no compile_opt: 7.9020839
compile_opt strictarr: 7.9031930
compile_opt strictarr, strictarrsubs: 7.8117480

So I see no slowdown, maybe even a speedup...

Cheers,

Brian

---------- ----------
Brian Larsen
Boston University
Center for Space Physics

Subject: Re: Invalid indices?
Posted by Michael Galloy on Thu, 29 Oct 2009 02:55:10 GMT
View Forum Message <> Reply to Message

Brian Larsen wrote:
> There are quite a few compile options out there, some I use normally
> and some I don't. THis second I am worrying about execution speed
> differences for compile options.
>
> I am tempted to use strictarrsubs and strictarr in all my codes,
> anyone have any thoughts on if there is a speed difference with any
> compile options? I will do a test also but I'm curious of others
> thoughts and experiences.
>
> Using this simple code I see:
> PRO run_test
> a1 = findgen(1000)
> a2 = findgen(1000)
> t0 = systime(/sec)
>   FOR i = 0UL, 1000 DO BEGIN
>   s3 = a1#a2

> ENDFOR
> print, systime(/sec)-t0
> END
>
> no compile_opt: 7.9020839
> compile_opt strictarr: 7.9031930
> compile_opt strictarr, strictarrsubs: 7.8117480
>
> So I see no slowdown, maybe even a speedup...

I use strictarr in everything I do because of subtle, difficult issues I have had in the past without it. I would be interested in the results of the time tests, but I wouldn't consider stopping use of it unless the results were very bad for it.

I use logical_predicate in special circumstances, but don't use any of the other options. I guess I just haven't been bitten by those particular problems yet.

Mike
--
www.michaelgalloy.com
Research Mathematician
Tech-X Corporation