Subject: Re: precedence question
Posted by JD Smith on Fri, 18 Mar 2005 23:45:54 GMT
View Forum Message <> Reply to Message

On Tue, 15 Mar 2005 14:55:16 +0100, Fï¿½ldy Lajos wrote:

>
> On Tue, 15 Mar 2005, m_schellens@hotmail.com wrote:
>
>> What do you want to hear?
>> Obviously the docmentation is wrong here and brackets have a higher
>> priority.
>> marc
>>
>
> Well, I'd like to know whether it is a software bug, a documentation bug,
> or simply I am reading something wrong. It is not obvious for me :-)
>
> You voted for documentation bug, thanks.
>
> David's Operator Precedence Tutorial has the same table, and refers to
> '[]' and '.' as equal precedence operators, so it is wrong, too :-)

Well, it's actually my tutorial David hosts, and I admit I stole the
precedence table straight from the manual without extensive
verification. That said, I think you guys have this whole issue
wrong.

You're indexing a 3D array using only two of three dimensions. Now
you might complain that IDL has irregular behavior in this case, but I
don't see it as a precedence issue. Consider instead:

```
IDL> a=replicate({l:randomu(sd,10)},5)
IDL> print,a[2].l[6]
   0.0162049
IDL> print,(a[2]).l[6]
   0.0162049
IDL> print,((a[2]).l)[6]
   0.0162049
```

Looks pretty good. In fact, if you think about it, there is no way
for `.' and `[]' to have anything but equal precedence. Let's say the
precedence of '[]' really was higher. How would IDL parse a[2].l[6]?
Let's see, a[2] is a scalar structure, but now we must first subscript
that with [6], since that has higher precedence.... hmmm.

What you've really run into is the apparently variable way IDL treats indexing expressions which specify some, but not all dimensions:

```
IDL> a=findgen(10,10,10)
IDL>  print,a[4,5:6,0]
      54.0000
      64.0000
IDL> print,a[4,5:6]
      54.0000
      64.0000
```

So here IDL just assumes you meant the first plane, as a convenience
to you. It could have thrown an error and said "Sorry try again
buddy", but it was being friendly instead. The only difference
between your case A and case C is that in case A IDL does it's
indexing/structure derefencing one by one and builds up the array from
the structure, whereas in case C you *first* construct the array of
size [2,3,4], and then index it with the incomplete index set [*,0:1],
at which point the "assume he meant the first plane" rule for arrays
kicks in, and you get the different result. By the way, if you had
used

```
IDL> help, a[3].l[*,0:1]
```

instead you would have found consistent behavior as well, since there
you're indexing only a 2x3 array.

JD