Subject: Re: When Ptr_New doesn't work
Posted by Malcolm Walters on Tue, 22 Jan 2002 14:51:46 GMT
View Forum Message <> Reply to Message

"Carles Badenes" <badenes@ieec.fcr.es> wrote in message
news:d46481f7.0201220342.3af528fa@posting.google.com...

```
>  I have the following  problem:
>
>    rCoefficients=PtrArr(nElems)
>    FOR i=0, nElems-1 DO BEGIN
>       rCoefficients[i]=Ptr_New(PtrArr(elems[i]))
>    ENDFOR
>
>  since elems is a vector of integers, each element i of
>  rCoefficients is a pointer to an array of elems[i] pointers.
>  But, for some reason,
>
>    ((*rCoefficients[j])[k]) = Ptr_New(FltArr(2), /NO_COPY)
>
>  doesn't work. k and j are within the allowed range, of course, and
>  ((*rCoefficients[j])[k]) is a null pointer, as expected from the
>  initialization above. Ptr_New is supposed to allocate memory for the
>  specified pointer to store an array of 2 floats, but I get the message
>
>    Expression must be named variable in this context: <POINTER
>  (<NullPointer>)>.
>
>  I must be doing something wrong. Can you help?
>
>  Thanks,
>  Carles
```

"Carles Badenes" <badenes@ieec.fcr.es> wrote in message
news:<d46481f7.0201220342.3af528fa@posting.google.com>...

```
>  I have the following  problem:
>
>    rCoefficients=PtrArr(nElems)
>    FOR i=0, nElems-1 DO BEGIN
>       rCoefficients[i]=Ptr_New(PtrArr(elems[i]))
>    ENDFOR
>
>  since elems is a vector of integers, each element i of
>  rCoefficients is a pointer to an array of elems[i] pointers.
>  But, for some reason,
>
>    ((*rCoefficients[j])[k]) = Ptr_New(FltArr(2), /NO_COPY)
>
>  doesn't work. k and j are within the allowed range, of course, and
```

> ((*rCoefficients[j])[k]) is a null pointer, as expected from the
> initialization above. Ptr_New is supposed to allocate memory for the
> specified pointer to store an array of 2 floats, but I get the message
>
>   Expression must be named variable in this context: <POINTER
> (<NullPointer>)>.
>
> I must be doing something wrong. Can you help?
>
This seems to be due to how IDL allocates and dereferences its pointers, consider the code below. The first part seems to be what you are trying to do, I have just expanded it.
This doesn't work since when you do the inner 'Ptr_new' 'tempVar' moves in memory. I guess the error was that this change could not occur in you condensed ((*rCoefficients[j])[k]) command.
The solution is to create the inner part and then set the pointer to it afterwards.

I hope this is of help
Malcolm Walters

```
PRO TEST

nElems=2
elems=[3,2,1]

rCoefficients=PtrArr(nElems)
FOR i=0, nElems-1 DO BEGIN
    rCoefficients[i]=Ptr_New(PtrArr(elems[i]))
    tempVar=(*rCoefficients[i])
    FOR j=0, elems[i]-1 DO BEGIN
     tempVar[j] = Ptr_New(FltArr(2), /NO_COPY)
     ; tempVar no longer equals *rCoefficients[i]
     (*tempVar[j])[0] = 100-i
     (*tempVar[j])[1] = 200-j
    ENDFOR
ENDFOR

rCoefficients=PtrArr(nElems)
FOR i=0, nElems-1 DO BEGIN
    tempVar=(PtrArr(elems[i]))
    FOR j=0, elems[i]-1 DO BEGIN
     tempVar[j] = Ptr_New(FltArr(2), /NO_COPY)
     ; tempVar no longer equals *rCoefficients[i]
     (*tempVar[j])[0] = 100-i
     (*tempVar[j])[1] = 200-j
    ENDFOR
    rCoefficients[i]=Ptr_New(tempVar)
ENDFOR


FOR i=0, nElems-1 DO BEGIN
 FOR j=0, elems[i]-1 DO BEGIN
  print,i,j,(*(*rCoefficients[i])[j])
 ENDFOR
ENDFOR

END
```

Subject: Re: When Ptr_New doesn't work
Posted by Richard Younger on Tue, 22 Jan 2002 17:36:22 GMT
View Forum Message <> Reply to Message

Hi, Carles.

There's something very fishy here. Your code seems perfectly correct to me. I found that when the outer set of parenthesis on the left hand side of the offending line of code is omitted, your program snippet seems to work fine. When I include the outer set of parenthesis, it gives me an error on versions Win 5.3-5.5, and possibly earlier. My test program is below. Either this is a bug or a very strange quirk of the language with obscure origins. Perhaps it should be reported?

For anyone else out there running on different systems, do other versions have this problem?

Best,
Rich

--

```
PRO test

   nElems = 4
   elems = intarr(nElems)+2
   rCoefficients=PtrArr(nElems)

   FOR i=0, nElems-1 DO BEGIN
      rCoefficients[i]=Ptr_New(PtrArr(elems[i]))
   ENDFOR

   FOR j=0, nElems-1 DO BEGIN
      FOR k=0, elems[j]-1 DO BEGIN
  ;Works:
        (*rCoefficients[j])[k] = Ptr_New(FltArr(2), /NO_COPY)

  ;Causes Error:
        ;((*rCoefficients[j])[k]) = Ptr_New(FltArr(2), /NO_COPY)
      ENDFOR
   ENDFOR

   ;;--- Clean up ----
   ;; - as in "I don't do any".

END
```

--
Richard Younger

Carles Badenes wrote:

```
>
>  I have the following  problem:
>
>    rCoefficients=PtrArr(nElems)
>    FOR i=0, nElems-1 DO BEGIN
>       rCoefficients[i]=Ptr_New(PtrArr(elems[i]))
>    ENDFOR
>
>  since elems is a vector of integers, each element i of
>  rCoefficients is a pointer to an array of elems[i] pointers.
>  But, for some reason,
>
>    ((*rCoefficients[j])[k]) = Ptr_New(FltArr(2), /NO_COPY)
>
>  doesn't work. k and j are within the allowed range, of course, and
>  ((*rCoefficients[j])[k]) is a null pointer, as expected from the
>  initialization above. Ptr_New is supposed to allocate memory for the
>  specified pointer to store an array of 2 floats, but I get the message
>
>    Expression must be named variable in this context: <POINTER
>  (<NullPointer>)>.
>
>  I must be doing something wrong. Can you help?
>
>                   Thanks,
>                                 Carles
```

Subject: Re: When Ptr_New doesn't work
Posted by badenes on Tue, 22 Jan 2002 17:38:00 GMT

View Forum Message <> Reply to Message

Thanks for that, Malcom!

I found an alternative solution:

```
  rCoefficients = PtrArr(nElems, /ALLOCATE_HEAP)
  FOR i = 0,  nElems-1 DO BEGIN
    *rCoefficients[i] = PtrArr(elems[i], /ALLOCATE_HEAP)
  ENDFOR
```

then allocate memory for each pointer

```
   (*(*rCoefficients[j])[k]) = FltArr(2)
```

The trick here is the /ALLOCATE_HEAP keyword, that allows the pointers
to be dereferenced. This works, but now I know why I was wrong before
;)

Subject: Re: When Ptr_New doesn't work
Posted by John-David T. Smith on Tue, 22 Jan 2002 20:02:18 GMT

View Forum Message <> Reply to Message

Richard Younger wrote:
>
> Hi, Carles.
>
> There's something very fishy here. Your code seems perfectly correct to
> me. I found that when the outer set of parenthesis on the left hand
> side of the offending line of code is omitted, your program snippet
> seems to work fine. When I include the outer set of parenthesis, it
> gives me an error on versions Win 5.3-5.5, and possibly earlier. My
> test program is below. Either this is a bug or a very strange quirk of
> the language with obscure origins. Perhaps it should be reported?
>
> For anyone else out there running on different systems, do other
> versions have this problem?

```
>         ;Works:
>         (*rCoefficients[j])[k] = Ptr_New(FltArr(2), /NO_COPY)
>
>         ;Causes Error:
>         ;((*rCoefficients[j])[k]) = Ptr_New(FltArr(2), /NO_COPY)
```

There's nothing mysterious about this error:

```
IDL> a=fltarr(4)
IDL> a[1]=1
IDL> (a[1])=1
% Expression must be named variable in this context: <FLOAT    (
  1.00000)>.
% Execution halted at: $MAIN$
```

The problem is with IDL's somewhat strict definition of what can consititute a "left-hand", or assignable, value. It handles the simple case of:

```
IDL> ((((a))))=1
```

without choking, but throw anything more complicated at the LHS, and it often falls down. It's a known weakness of IDL, revealed in a complicated way. ;) One good rule of thumb is that, if the LHS contains subscripts, the last character before the "=" (modulo whitespace), should be "]" (ok Craig, or ")").

Here's another example:

```
IDL> a={a:1}
IDL> a.a=1
IDL> (a.a)=1
% Expression must be named variable in this context: <INT      (
1)>.
% Execution halted at: $MAIN$
```

The same rule applies. If structure dereference occurs, the sequence prior to "=" must be an unadulterated tag (unless you're combining structure dereference and subscripting... it can get ugly).

I like to work from the inside out with long nested *[]() sequences. A really nice feature to be released in an upcoming version of IDLWAVE lets you do in-place inspection of parts of the command line (or prior commands visible in the buffer) while your composing it, similar to the Shift-click and Shift-Drag printing available now in the buffer. For instance, while composing:

```
(*rCoefficients[j])[k]=
```

you could key-drag various regions (or trust IDLWAVE's ability to pick the expression of the "right" length nearby), and inspect them with customized commands ("print,size(___,/DIMENSIONS)" being one of my current favorites).

This feature helps a lot with these kinds of deeply nested compositions (even in the buffer! -- just set a breakpoint and inspect away). It

would also help if RSI would generate a true operator precedence table
including "[]" (array subscript) and "." (structure dereference).

Good luck,

JD

Subject: Re: When Ptr_New doesn't work
Posted by badenes on Wed, 23 Jan 2002 11:43:07 GMT
View Forum Message <> Reply to Message

Well, thanks everybody.
 I'll look forward to that command line editing feature for IDLWAVE.
In fact, the reasons I had the code so cluttered with parentheses were
A) to avoid making assumptions about the precedence rules I don't know
and B) to help me while debugging with IDLWAVE. The parentheses let me
pick any chunk of the nested dereferences and display the offending
pointer or array. That feature you mentioned would make things even
easier.
 Cheers,
  Carles

Subject: Re: When Ptr_New doesn't work
Posted by Richard Younger on Wed, 23 Jan 2002 17:27:17 GMT
View Forum Message <> Reply to Message

JD Smith wrote:

[...]

> The problem is with IDL's somewhat strict definition of what can
> consititute a "left-hand", or assignable, value. It handles the simple
> case of:
>
> IDL> ((((a))))=1
>
> without choking, but throw anything more complicated at the LHS, and it
> often falls down. It's a known weakness of IDL, revealed in a
> complicated way. ;) One good rule of thumb is that, if the LHS contains
> subscripts, the last character before the "=" (modulo whitespace),
> should be "]" (ok Craig, or ")").

[...]

This seems very strange to me. I think of parenthesis as not modifying

their arguments at all, since they should (in my own little idl, er,
ideal world) change only precedence and not value. Go figure. As you
say, it must be fallout from dynamic typing and/or that incomplete
operator precidence.

I am in the habit of adding parens everywhere I can when constructing
one of those long complicated lines of code from the inside out. I then
make sure the thing works, start from the outside in and remove those
that seem unneccessary. I'm completely suprised I haven't been
frustrated by this quirk before. I guess I must have never put an
outside pair on a LHS, or never noticed the error for something else.

Thanks for clearing things up, JD. It's nice to know why things are
screwy, even if it is "because that's the way they are."

Best,
Rich

--
Richard Younger   MIT Lincoln Laboratory
Phone: (781)981-4464   244 Wood St.
Fax: (781)981-0122   Lexington, MA 02421