Subject: Null terminated strings
Posted by James Kuyper on Mon, 07 Jan 2002 18:11:52 GMT
View Forum Message <> Reply to Message

I'm reading a string-valued file attribute from an HDF file that was created using C code. As seems quite reasonable for C programs, the attribute was written with a length that includes a terminating null character. When I read it in using IDL, that null character got included as well. This causes a number of bizarre effects, most notably:

IDL> print,date
2001-10-07
IDL> print,date+'T12:00:00'
2001-10-07'T12:00:0

I can handle this particular case by using strmid(date,0,10), but in general a file attribute might contain multiple null-delimited strings, of unknown length. Is there an efficient way of converting such a string into an IDL string array?

Subject: Re: Null terminated strings
Posted by Jaco van Gorkom on Tue, 08 Jan 2002 16:17:27 GMT
View Forum Message <> Reply to Message

James Kuyper wrote:
> I'm still wonder how to best convert a null-delimited list of strings
> into an IDL string array (it's just curiousity, I don't have any
> immediate need for that ability).

Doesn't STRSPLIT(/EXTRACT) satisfy your curiosity?

Jaco

--
Jaco van Gorkom
FOM-Instituut voor Plasmafysica "Rijnhuizen", The Netherlands
e-mail: gorkom@rijnh.nl

Subject: Re: Null terminated strings
Posted by thompson on Tue, 08 Jan 2002 17:17:13 GMT
View Forum Message <> Reply to Message

James Kuyper <kuyper@gscmail.gsfc.nasa.gov> writes:

> Craig Markwardt wrote:

>> James Kuyper <kuyper@gscmail.gsfc.nasa.gov> writes:
>>
>>
>>> I'm reading a string-valued file attribute from an HDF file that was
>>> created using C code. As seems quite reasonable for C programs, the
>>> attribute was written with a length that includes a terminating null
>>> character. When I read it in using IDL, that null character got included
>>> as well. This causes a number of bizarre effects, most notably:
>>>
>>> IDL> print,date
>>> 2001-10-07
>>> IDL> print,date+'T12:00:00'
>>> 2001-10-07'T12:00:0
>>>
>>> I can handle this particular case by using strmid(date,0,10), but in
>>> general a file attribute might contain multiple null-delimited strings,
>>> of unknown length. Is there an efficient way of converting such a string
>>> into an IDL string array?
>>
>>
>> What happens when you swizzle it through a STRING-BYTE-STRING
>> transformation?
>>
>> I.e.,
>>
>>   date = string(byte(date))
>>
>> I believe that STRING will ignore any trailing 0-bytes, hence this may
>> solve your problem exactly, at the expense of some extra CPU.

> Thanks - that worked. It only solves the single-string case, but that's
> the case I am currently facing. It saves me the trouble of figuring out
> how long the string is, and it does the right thing, whether or not the
> string is null-terminated.

> I'm still wonder how to best convert a null-delimited list of strings
> into an IDL string array (it's just curiousity, I don't have any
> immediate need for that ability). My best solution so far is to convert
> it to a byte array, find the null delimiting characters with where(),
> and then write a loop to convert each subarray into a seperate IDL
> string. This should work, but I'm always suspicious of the efficiency of
> any solution for an IDL problem that involves an explicit loop.

As far as I can determine, that should work equally as well with arrays as
with strings. For example,

```
IDL> test = ['This','is','a','test']
IDL> btest=byte(test)
IDL> print,btest
  84 104 105 115
 105 115   0   0
  97   0   0   0
 116 101 115 116
IDL> stest = string(btest)
IDL> help,stest
STEST           STRING    = Array[4]
IDL> print,strlen(stest)
          4           2           1           4
IDL> print,stest
This is a test
```

You shouldn't have to use a loop.

Bill Thompson

Subject: Re: Null terminated strings
Posted by Craig Markwardt on Tue, 08 Jan 2002 19:51:50 GMT
View Forum Message <> Reply to Message

James Kuyper <kuyper@gscmail.gsfc.nasa.gov> writes:
> Craig Markwardt wrote:
...
>> What happens when you swizzle it through a STRING-BYTE-STRING
>> transformation?
>>
>> I.e.,
>>
>> date = string(byte(date))
>>
>> I believe that STRING will ignore any trailing 0-bytes, hence this may
>> solve your problem exactly, at the expense of some extra CPU.
>
> Thanks - that worked. It only solves the single-string case, but that's
> the case I am currently facing. It saves me the trouble of figuring out
> how long the string is, and it does the right thing, whether or not the
> string is null-terminated.
>
> I'm still wonder how to best convert a null-delimited list of strings
> into an IDL string array (it's just curiousity, I don't have any
> immediate need for that ability). My best solution so far is to convert
> it to a byte array, find the null delimiting characters with where(),
> and then write a loop to convert each subarray into a seperate IDL
> string. This should work, but I'm always suspicious of the efficiency of

> any solution for an IDL problem that involves an explicit loop.

The problem is that IDL has no way to represent the 0th ASCII
character in a string. At least no way that I can find, other than
bringing the data in from outside, as you have done with HDF.

My best solution is to do as you have, which is to convert to BYTEs,
then locate the 0's. But at this stage you can quickly replace the
0's with some other control character, say ASCII 1. [This assumes
that 1 = CTRL-A never appears in your strings.] Then you can convert
back to string and use STRSPLIT or STR_SEP to split it up.

However, you should be aware that STR_SEP uses FOR loops, and I have
never really noticed an impact when I've used it. Unless your code is
actually a dog with FOR loops, versus *hypothetically* a dog, then
there is no reason to optimize it. :-)

Craig

--
-- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
-- --------------

Subject: Re: Null terminated strings
Posted by James Kuyper on Tue, 08 Jan 2002 20:58:53 GMT
View Forum Message <> Reply to Message

William Thompson wrote:

> James Kuyper <kuyper@gscmail.gsfc.nasa.gov> writes:
...

>> I'm still wonder how to best convert a null-delimited list of strings
>> into an IDL string array (it's just curiousity, I don't have any
>> immediate need for that ability). My best solution so far is to convert
>> it to a byte array, find the null delimiting characters with where(),
>> and then write a loop to convert each subarray into a seperate IDL
>> string. This should work, but I'm always suspicious of the efficiency of
>> any solution for an IDL problem that involves an explicit loop.
>
>
>
> As far as I can determine, that should work equally as well with arrays as
> with strings. For example,
>

> IDL> test = ['This','is','a','test']
> IDL> btest=byte(test)
> IDL> print,btest
> 84 104 105 115
> 105 115 0 0
> 97 0 0 0
> 116 101 115 116
> IDL> stest = string(btest)
> IDL> help,stest
> STEST STRING = Array[4]
> IDL> print,strlen(stest)
> 4 2 1 4
> IDL> print,stest
> This is a test
>
> You shouldn't have to use a loop.

You're relying there on the fact that btest is a two-dimensional array; string() converts each row into a seperate string. In the case I'm worrying about, I would have a single IDL string, containing null characters at arbitrary positions. Try the following:

btest = [84B,104B,105B,115B,0B,105B,115B,0B,97B,0B,106B,101B,115B,11 6B]
stest = string(btest)

All you get in stest is the 'This'.

Subject: Re: Null terminated strings
Posted by Struan Gray on Wed, 09 Jan 2002 08:29:02 GMT
View Forum Message <> Reply to Message

James Kuyper, kuyper@gscmail.gsfc.nasa.gov writes:

> I'm still wonder how to best convert a null-delimited
> list of strings into an IDL string array (it's just
> curiousity, I don't have any immediate need for that
> ability). My best solution so far is to convert it to a byte
> array, find the null delimiting characters with where(), and
> then write a loop to convert each subarray into a seperate
> IDL string. This should work, but I'm always suspicious of
> the efficiency of any solution for an IDL problem that
> involves an explicit loop.

</lurk>

Sounds like a job for supersonic HISTOGRAM and his ever-eager sidekick REVERSE_INDICES.

Struan
<lurk>

Subject: Re: Null terminated strings
Posted by Malcolm Walters on Wed, 09 Jan 2002 10:26:23 GMT
View Forum Message <> Reply to Message

"James Kuyper" <kuyper@gscmail.gsfc.nasa.gov> wrote in message
news:3C3B5D8D.9050806@gscmail.gsfc.nasa.gov...
>
> You're relying there on the fact that btest is a two-dimensional array;
> string() converts each row into a seperate string. In the case I'm
> worrying about, I would have a single IDL string, containing null
> characters at arbitrary positions. Try the following:
>
> btest = [84B,104B,105B,115B,0B,105B,115B,0B,97B,0B,106B,101B,115B,11 6B]
> stest = string(btest)
>
> All you get in stest is the 'This'.
>

Personally I would do this in two stages,
Firstly if you replace each 0B with a 10B using
   btest[where(btest eq 0B)]=10B
you can now print btest then you get the line breaks (This can be used in
message boxes etc.)
Or if you want to go one further then split them using
   stest = strsplit(string(btest),string(10B),/extract)
   print, stest
      This is a jest
   help,stest
      <Expression>   STRING   = Array[4]
I hope this is of some help.
Malcolm Walters

Subject: Re: Null terminated strings
Posted by James Kuyper on Wed, 09 Jan 2002 14:19:56 GMT
View Forum Message <> Reply to Message

Malcolm Walters wrote:
...

> Firstly if you replace each 0B with a 10B using

> btest[where(btest eq 0B)]=10B
> you can now print btest then you get the line breaks (This can be used in
> message boxes etc.)
> Or if you want to go one further then split them using
> stest = strsplit(string(btest),string(10B),/extract)
> print, stest
> This is a jest
> help,stest
> <Expression> STRING = Array[4]
> I hope this is of some help.
> Malcolm Walters

Thanks! That sounds like exactly what I was looking for. I'm still
enough of a newbie to have been unaware of the existence of strsplit().

Subject: Re: Null terminated strings
Posted by Craig Markwardt on Wed, 09 Jan 2002 23:43:29 GMT
View Forum Message <> Reply to Message

Struan Gray <struan.gray@sljus.lu.se> writes:

> James Kuyper, kuyper@gscmail.gsfc.nasa.gov writes:
>
>> I'm still wonder how to best convert a null-delimited
>> list of strings into an IDL string array (it's just
>> curiousity, I don't have any immediate need for that
>> ability). My best solution so far is to convert it to a byte
>> array, find the null delimiting characters with where(), and
>> then write a loop to convert each subarray into a seperate
>> IDL string. This should work, but I'm always suspicious of
>> the efficiency of any solution for an IDL problem that
>> involves an explicit loop.
>
> </lurk>
>
> Sounds like a job for supersonic HISTOGRAM and his
> ever-eager sidekick REVERSE_INDICES.

That's a good idea, although I think you can't avoid a FOR loop. In
fact, it is my belief that by using REVERSE_INDICES to look at more
than one bin in a histogram, you are *guaranteed* to use a FOR loop or
equivalent.

Craig

Subject: Re: Null terminated strings
Posted by Struan Gray on Thu, 10 Jan 2002 14:57:19 GMT
View Forum Message <> Reply to Message

Craig Markwardt, craigmnet@cow.physics.wisc.edu writes:
> Struan Gray <struan.gray@sljus.lu.se> writes:
>> James Kuyper, kuyper@gscmail.gsfc.nasa.gov writes:
>>
>>> I'm still wonder how to best convert a null-delimited
>>> list of strings into an IDL string array
>>
>> Sounds like a job for supersonic HISTOGRAM and his
>> ever-eager sidekick REVERSE_INDICES.
>
> That's a good idea, although I think you can't avoid a FOR loop. In
> fact, it is my belief that by using REVERSE_INDICES to look at more
> than one bin in a histogram, you are *guaranteed* to use a FOR loop or
> equivalent.

Aahh. But in this case, you are only going to look at one bin, and the first one at that - which avoids the usual problem of having to step through the REVERSE_INDICES array. I haven't tried it, but it might even be possible to force Histogram to construct a histogram with just one bin. Then you're laughing.

Of course, this is much the same as using where(), except that as those who have read the HISTOGRAM documentation know, it's faster doing it the non-obvious way.

Mind you, on further reflection, I would probably just adapt Malcolm Walters' idea to use array compares, which in theory at least should be faster than either HISTOGRAM or WHERE.

```
btext = byte(text)
btext = btext + 10B*(btext < 1B)
textarr = strsplit(string(btext), string(10B), /extract)
```

Struan