

---

Subject: Re: Null terminated strings

Posted by [James Kuyper](#) on Tue, 08 Jan 2002 15:49:51 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Craig Markwardt wrote:

```
> James Kuyper <kuyper@gsfmail.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 curiosity, 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 separate 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.

---