Subject: Concatenate elements of string array
Posted by kluegel on Mon, 09 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Does anyone out there have a clever idea on how to
concatenate all elements of a one dimensional array
of strings? Of course, explicit looping is the
obvious way to do it, but approaches using array
manipulation tend to be more efficient. Any
array-based approaches out there?

Thanks for any ideas.

-- Tom Kluegel
   Los Alamos National Lab
   Los Alamos, New Mexico, USA

Subject: Re: Concatenate elements of string array
Posted by kluegel on Thu, 12 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <7othpe$cin$1@nnrp1.deja.com>,
  kluegel@lanl.gov wrote:
> result = STRING(string_array, FORMAT='(16(16384(16384A)))')

I noticed that this approach still has problems with any nul characters
in array of strings. We will still have to make a rule that the input
array contains no strings with nul characters.

-- Tom

Subject: Re: Concatenate elements of string array
Posted by kluegel on Thu, 12 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

In article <7othpe$cin$1@nnrp1.deja.com>,
  kluegel@lanl.gov wrote:

```
>  result = STRING( string_array, FORMAT='(16(16384(16384A)))' )
```

One gotcha still appears to be nul characters, which gives weird
behavior. This is likely because IDL is implemented in C/C++ which uses
nul terminated strings. So we still have to impose the rule that no
nul characters are allowed in any strings in the string array.

-- Tom Kluegel

Sent via Deja.com http://www.deja.com/
Share what you know. Learn what you don't.

Subject: Re: Concatenate elements of string array
Posted by kluegel on Thu, 12 Aug 1999 07:00:00 GMT

View Forum Message <> Reply to Message

In article <37B1730A.41C6@icg.tu-graz.ac.SIG>,
  Ruhaltinger Norbert <norbertr@icg.tu-graz.ac.SIG> wrote:

```
>  string can be used like sprintf in C
>  You have to create the Format specification on the fly
>
>  IDL> string_array=['one ','string ','split ',$
>    'into ','many ','array elements']
>  IDL> help,string_array & print,string_array
>  STRING_ARRAY    STRING    = Array[6]
>  one  string  split  into  many  array elements
>  IDL> one_string = string(string_array, $
>    format = '(' + string(n_elements(string_array)) + 'A)')
>
>  IDL> help,one_string & print,one_string
>  ONE_STRING      STRING    = 'one string split into many array
elements'
>  one string split into many array elements
```

Thank you, Norbert!

You gave me the idea that I can use a format that isn't variable at all,
but just larger than I'll ever need:

```
result = STRING( string_array, FORMAT='(16(16384(16384A)))' )
```

I ran some benchmarks which showed that the above approach is more than
20 times faster (Pentium II 400) than the more obvious explicit looping
algorithm:

```
result = ''
```

```
FOR i=0, N_ELEMENTS(string_array)-1 DO result = result + string_array[i]
```

As usual, when we avoid explicit looping, etc. in interpreted code we get a decent speedup. For greater generality, I made it capable of handling very long arrays. The new approach allows an array of up to 2^32 elements. It was necessary to nest the repeat counts in order to get around IDL's maximum of 32767. Note that the only way such a large array could be used would be if many elements were empty strings. Otherwise the result would be a string longer than 32767 characters, IDL's limit for string length.

I think this is a bit kludgey, but its so much faster than looping I'll live with it!

Thank you everyone who took the time to brainstorm this little puzzle with me.

-- Tom Kluegel