Subject: Re: IDL 6.2 Array Definition Question (Program code area full)
Posted by mshep on Wed, 19 Apr 2006 17:49:04 GMT
View Forum Message <> Reply to Message

Hi Reimar,

The main question is why does IDL place a limit on the amount of array
elements being passed explicitly?

help, !version, /str

```
IDL> help, !version, /str
** Structure !VERSION, 8 tags, length=104, data length=100:
   ARCH            STRING    'x86_64'
   OS              STRING    'linux'
   OS_FAMILY       STRING    'unix'
   OS_NAME         STRING    'linux'
   RELEASE         STRING    '6.2'
   BUILD_DATE      STRING    'Jun 20 2005'
   MEMORY_BITS     INT             64
   FILE_OFFSET_BITS
                   INT             64
IDL>
```

As for the reason why I am passing this many elements explicitly is
that the arrays are being generated in a perl script and then passed to
an IDL program.

Below is a simple example where I am explictly passing arrays from perl
to IDL, but I run into a limitation on the amount of elements that I
can pass explicitly:

```
#!/usr/bin/perl -w

@a_arr =
 (0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9 ,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9);
@b_arr =
 (0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9 ,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9);
@c_arr =
 (0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9 ,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9);
@d_arr =
 (0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9 ,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9);
@e_arr =
 (0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9 ,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9);
@f_arr =
 (0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9 ,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9);
@g_arr =
 (0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9 ,0,1,2,3,4,5,6,7,8,9,0,1,2,3,4,5,6,7,8,9);
```

```
$str_a =  "@a_arr";          # put array into a string with blanks
between elements
$str_a =~ s/ /','/g;         # substitute
$str_a = "['".$str_a."']"; # Add IDL brackets and end quotes
$str_b =  "@b_arr";          # put array into a string with blanks
between elements
$str_b =~ s/ /','/g;         # substitute
$str_b = "['".$str_b."']"; # Add IDL brackets and end quotes
$str_c =  "@c_arr";          # put array into a string with blanks
between elements
$str_c =~ s/ /','/g;         # substitute
$str_c = "['".$str_c."']"; # Add IDL brackets and end quotes
$str_d =  "@d_arr";          # put array into a string with blanks
between elements
$str_d =~ s/ /','/g;         # substitute
$str_d = "['".$str_d."']"; # Add IDL brackets and end quotes
$str_e =  "@e_arr";          # put array into a string with blanks
between elements
$str_e =~ s/ /','/g;         # substitute
$str_e = "['".$str_e."']"; # Add IDL brackets and end quotes
$str_f =  "@f_arr";          # put array into a string with blanks
between elements
$str_f =~ s/ /','/g;         # substitute
$str_f = "['".$str_f."']"; # Add IDL brackets and end quotes
$str_g =  "@g_arr";          # put array into a string with blanks
between elements
$str_g =~ s/ /','/g;         # substitute
$str_g = "['".$str_g."']"; # Add IDL brackets and end quotes

# Use a here document to run IDL from linux/unix command line

$idl_prog_full_test = "full_test";
$idl_run = ".r $idl_prog_full_test";
$exit = "exit";
$idl_command_try_2 = "$idl_prog_full_test, $str_a, $str_b, $str_c,
$str_d, $str_e, $str_f, $str_g";
system("idl
<<IDL_INPUT\n$idl_run\n$idl_command_try_2\n$exit\nIDL_INPUT\n ");

exit;
```