Subject: Re: String array from C
Posted by Allan Whiteford on Fri, 06 Feb 2009 16:58:21 GMT
View Forum Message <> Reply to Message

Robbie wrote:
> I've had some problems dealing with string arrays which have been
> generated from a C program.
>
> If I try LONG(str) or FIX(str) I will find that there are extra
> characters which get processed, yet I don't see them when I try PRINT
> and HELP.
> I can workaround the problem by using LONG(STRMID(0,str)) and FIX
> (STRMID(0,str)).
>
> Perhaps I have string processing mixed up. But, I thought that
> IDL_StrEnsureLength took are of everything. I've changed my C code and
> the problem has gone away, but I still don't understand why.
>
> The original C code was:
> char* v = PQgetvalue(Res,row,col);
> int vl = strlen(v);
> IDL_StrEnsureLength(&(pisArray[i]), vl);
> strncpy(pisArray[i].s, v, vl);
>
> The new C code is:
> char* v = PQgetvalue(Res,row,col);
> int vl = strlen(v);
> IDL_StrEnsureLength(&(pisArray[i]), vl+1);
> strncpy(pisArray[i].s, v, vl+1);
> pisArray[i].slen = vl;
>
> Thanks
> Robbie

Hi,

You're supposed to use:

IDL_StrStore

to copy a string from C-land to IDL-land. If you don't then you'll end
up in a whole world of hurt.

This will take care of setting the slen part of the IDL_STRING
structure. You can't rely on the null termination character from C
meaning anything to IDL.

Also, if you write directly into the .s pointer (which points to the

actual string) there is a chance you'll overwrite totally unrelated strings. For efficiency, if you have two string variables set to the same value then IDL will (sometimes!) only store the data once in memory and point both the .s pointers at the same place. IDL keeps track of when the pointers need to point to different places when one of the strings changes (as does IDL_StrStore) but if you mess with the data yourself directly then you can end up modifying variables you never intended to.

Thanks,

Allan
