Subject: Re: HANDLE_FREE: when to use? is it necessary?
Posted by chase on Mon, 29 Jul 1996 07:00:00 GMT
View Forum Message <> Reply to Message

>>>> > "Stein" == Stein Vidar (UiO) <steinhh@cdsa2.nascom.nasa.gov> writes:
In article <4tb4cj$dq7@post.gsfc.nasa.gov> steinhh@cdsa2.nascom.nasa.gov (Stein Vidar (UiO)) writes:

Stein> In article <Pine.SOL.3.91.960722114953.671B-100000@mapper.mbt.washington.edu>,
Russ Welti <rwelti@mapper.mbt.washington.edu> writes:
Stein> |>
Stein> |> I was told once by RSI that if one is only about to reassign
Stein> |> a handle to point to a NEW piece of data (when it already
Stein> |> points to some other data) it is *not* necessary to call
Stein> |> HANDLE_FREE.
Stein> |>

This makes sense. In this context one can view handles like other IDL variables. You do not need to free a variable's storage (e.g. by using the temporary() command) every time you assign a new value to the variable. Similarly, the freeing of storage a handle points to happens automatically. So handle_free is unnecessary when you only want to change the handle's value.

Stein> When I started using handles for my own purposes, I did a (too?)
Stein> quick check to se what effects handle_free could have. As it turns
Stein> out, the handle *numbers* appeared to be always increasing regardless
Stein> of handle_free calls, so I figured that RSI just hadn't gotten around
Stein> to actually freeing any handles.

The handles do appear to get freed (as revealed by handle_info() and help,/handle). Just because the IDs returned by handle_create() are not immediately reused does not mean the handle is not freed.

I use IDL version 4.0.1 on various UNIX systems (HPUX, IRIX, SunOS). "help,/handle" gives some revealing statistics about handles. It seems that IDL keeps track of handle references using a small hash table (containing only 421 slots on our systems, but the table probably only holds top level handles). Using your test program shows that the table does not grow when using handle_free. However, the hash table quickly becomes overwhelmed when handle_free is not used and you create thousands of unique handles. When there are thousands of active handles chaining is used off of each table slot making the table very inefficient when you call handle_create (probably for all other handle functions too).

Because top-level handles are like a global variable there is really no reason to have thousands of them available simultaneously - it would

be like a C program with thousands of different variables which would overwhelm most C compilers.

Stein> |> If freeing handles is really the answer, and is a required practise,
Stein> |> then why so few occurrences of its use in IDL programs?
Stein> |>

Use handle_free whenever you will no longer need the handle, e.g., when exiting the scope of the reference to the handle. Here is where the real problem of using handles can get you. If you fail to free the handle value storage and lose the handle ID then you end up with a memory leak problem like those that plague C and C++ programmers. IDL can not know if you have references to a handle since handle IDs are represented as ordinary long integers rather then by a unique type. Thus freeing the memory must be done manually by the user.

Chris

--
==========
Bldg 24-E188
The Applied Physics Laboratory
The Johns Hopkins University
Laurel, MD 20723-6099
(301)953-6000 x8529
chris.chase@jhuapl.edu