Subject: Finding Memory Leak ?
Posted by Richard Tyc on Thu, 21 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

I am starting to lose hair over this one. I recently noticed that when I close my program, two pointer variables remain which I am reluctant to remove using PTR_FREE, PTR_VALID()

I have checked every line of code where I create a new variable using PTR_NEW and I always properly used PTR_Free on it. I also use the DICOM method GetValue alot (which returns a pointer to the data) but always use /NO_COPY so the pointers point to actual data within the object which should get properly removed when the object is destroyed.

Anyone care to give any advice how I can find the source of the bug ? I have been trying to print out all the valid pointers (ie. print, PTR_VALID()) between Function/Procedure calls to determine when the 2 pointers come alive but this is very tedious (and so far unsuccessful) !!

Thanks
--
Richard Tyc
Project Engineer
St. Boniface Hospital Research Center
351 Tache Ave
Winnipeg, MB
Canada
Tel: 204-237-2557
Fax: 204-231-0485
Email: richt@sbrc.umanitoba.ca

Subject: Re: Finding Memory Leak ?
Posted by promashkin on Fri, 22 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Echoing what David said, my chase after the ghostfont object immediately came to my mind. I used Randall's suggestion to verify it was a bug. The advice was to do the following.
As heap variables and pointers are created, the pointers (or orefs) are created sequentially. If you use Help, /heap before you kill the program, you will get the list of all existing heap references.
Copy-paste that to your favorite MS Word processor. Then, exit the widget program and get heap info on the leaking pointers. Compare pointer numbers with those you had in the program (using word processor allows faster search if you had a lot of them, as I did). Thus, you can locate when did they get created, and from that, who actually made them (you or RSI :-).

The ghostfont bug, actually, was not caused by any object alone. It was caused by something in .Reset_session that was not doing what it should have to the graphics system; I am not too good at that, maybe somebody from RSI could tell us better. The bug only appeared after using .Reset_session.
Cheers,
Pavel
