Subject: Re: Garbage collection and Memory
Posted by eharold on Fri, 10 Jun 1994 14:44:29 GMT
View Forum Message <> Reply to Message

|> In article <thompson.770745164@serts.gsfc.nasa.gov>, thompson@serts.gsfc.nasa.gov (William Thompson) writes:
|> |> hevans@estwm0.wm.estec.esa.nl (Hugh Evans) writes:
|> |>
|> |> >I have discovered that after using Wave for an extended period that it slowly
|> |> >grabs more and more memory, even if new variables are not created, until
|> |> >finally it runs out of core memory. Whereas by saving the session and
|> |> >restarting it, the previous operation that crashed on a memory allocation
|> |> >problem will complete successfully.
|> |>

I think I've bumped across this one myself and although the problem is documented, it's still nasty. Consider the following sequence

a=indgen(really_big_number)
a=findgen(some_other_number)

The allocation from the first statement isn't freed when the second assignment to a is made, even though, near as I can tell, there's no way to get at the values stored in a from the first assignment. On the other hand the sequence of statements

a=indgen(really_big_number)
a=0
a=findgen(some_other_number)

will free the first allocation. Why does assignment to a scalar free the memory allocated to a but assignment to an array just leaves it hanging? I have no idea, but this is how IDL works and it's quite annoying. Since after the a=findgen(some_other_number) there's no way to get at the original contents of a I see no reason not to automatically free a before making the assignment, but currently that's not what happens. Consequently unless you're careful to manully free arrays by setting them to scalars before reassigning them, you'll pile up a lot of allocated but unused memory.

--

Elliotte Rusty Harold National Solar Observatory
eharold@sunspot.noao.edu Sunspot NM 88349