Subject: Re: pointer question
Posted by Jaco van Gorkom on Thu, 22 Mar 2001 14:01:32 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
>
> "Ted Graves" <egraves@socrates.Berkeley.EDU> wrote in message
> news:99blck$ko7$1@agate.berkeley.edu...
...
>> result = TEST(PTR_NEW(value))
>>
>> where value is whatever i want the heap variable to be. What happens toth e
>> heap variable assigned in this statement after TEST returns? I'm assuming
>> from that because of the way it was created, a heap variable now exists that i
>> can't easily get rid of without using HEAP_GC.
>
> Yes.
>
> But if you have access to the code of TEST you could do this:
>
> pro test, a
>
> ; Do something with a
>
> if not arg_present(a) then if ptr_valid(a) then ptr_free, a
>
> end

Very nice! However, what if you pass in 'a' by value, e.g., from an
array of pointers?
If I call test like
for i=0, n_elements(PointerArray)-1 do test(PointerArray[i])
then I lose all the heap variables, right?

I would prefer to avoid the problem altogether by making TEST accept
both pointers and values, something like:

```
pro test, a
   if size(a, /type) ne 10 then begin
      a = ptr_new(a, /no_copy)
      a2ptr = 1
   endif else a2ptr = 0

   ; Do something with a, pointer-based.

   if a2ptr and ptr_valid(a) then begin
      a_copy = a
      a = temporary(*a)
```

```
        ptr_free, a_copy
    endif
end
```

If TEST is not your own code, this could easily be done in a wrapper routine as well. The flexibility of not having to bother about pointers-or-not is great for command-line use. But then again, using heap_gc on the command line every once in a while is not a big problem either...

  Jaco

Jaco van Gorkom   gorkom@rijnh.nl
FOM-Instituut voor Plasmafysica "Rijnhuizen", The Netherlands