
Subject: Re: Reading files with unknown amount of data

Posted by [zawodny](#) on Thu, 04 Nov 1993 12:44:42 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <2bao2r\$qi9@mailgzzr.TU-Berlin.DE> brau1231@camillo.fb12.tu-berlin.de writes:

> In article <28fbv1\$qq6@reznor.larc.nasa.gov>, [zawodny@arbd0.larc.nasa.gov](#) (Dr Joseph M Zawodny) writes:

> |> >Robert Davis ([rdavis@aerospace.aero.org](#)) wrote:

> |> >I am trying to read a data file containing an unknown amount of data into

> |> >arrays in IDL. I know the format of the data in the file, but not the number

> |> >of pieces of data in the file. Currently, I read the file twice; once to

> |> >determine the amount of data in the file and then a second time to actually

> |> >read the data into an array (now that I know the size of array needed).

> |> >Is there a better way to do this (without having to read the file twice)?

> |>

> |> Use the EOF function in IDL. Try something like this:

> |>

> |>

>

>

> That's good, but you can also add entry's to a array.

> IDL let grow the array automatically, too.

>

>

>

> array=0. ; to initialize a variable called array

>

>

> openr,lun,filename, /get_lun ; no interest of 'lun's name

> point_lun,lun,0 ; but 'lun' know what I mean

> ; (logical unit number)

>

> while not eof(lun) do begin

>

> ;I use mostly

> readf,lun,a ; reading the entry of 'lun'

>

> array=[array,a] ; idl set the variable 'array' (if existing)

> ; to the right dimensions / append one entry

> ; so-called 'a'

> endwhile

> close,lun

> free_lun,lun

>

>

> ;cut the first entry of the array (array=0.)

>

> array=(1:*)

>
> from now on, the array 'array' has all entries and the right dimension.
>
> --
> Rainer Brauckhoff TU-Berlin Germany
>
> E-Mail: brau1231@camillo.fb12.tu-berlin.de

If I understand the way IDL uses memory, this "growing" of the array could be a real memory hog. As I understand it, IDL will go off and try to find a contiguous piece of memory for the new array. The areas where the old ARRAYS will not be big enough for the new version of ARRAY. So if the final size of ARRAY is 10000 points, then this method will consume $(n*(n+1)/2)$ or 50 million points worth of space (200 MB if we are talking floating point numbers) in the process. Obviously this is a worse case scenario and is dependant upon what else you may be doing in your program. In my mind it is clearly more efficient to make an ARRAY which is way too big (like $[file\ size\ in\ bytes]/4$ elements).

--

Joseph M. Zawodny (KO4LW)
Internet: zawodny@arbd0.larc.nasa.gov
Packet: ko4lw@n4hog.va.usa

NASA Langley Research Center
MS-475, Hampton VA, 23681-0001