Subject: Re: Declaring large vectors in IDL
Posted by penteado on Tue, 20 Apr 2010 00:11:29 GMT
View Forum Message <> Reply to Message

On Apr 19, 8:57 pm, fgg <fabioguimaraesgoncal...@gmail.com> wrote:
> And here's the adapted script in case it helps:
>
> filters = ['*.txt', '*.dat', '*.out']
> infile = dialog_pickfile(/read, filter=filters)
> n = file_lines(infile)
> raw_data = strarr(n)
> openr, unit, infile, /get_lun
> readf, unit, raw_data
> close, unit & free_lun, unit
> datas = where(stregex(raw_data,'=',/
> boolean),ndata,complement=extra_lines)
> data = raw_data[datas]
> if (ndata lt n_elements(raw_data)) then begin
> extra_assoc = value_locate(datas,extra_lines)
> for i=0L,n_elements(extra_lines)-1 do $
> data[extra_assoc[i]] = strjoin([temporary(data[extra_assoc[i]]),$
> raw_data[extra_lines[i]]],/single)
> endif
> outfile = dialog_pickfile(/write, default_extension='pro',
> filter='*.pro')
> openw, outunit, outfile, /get_lun
> for i=0,ndata-1 do begin
> line = strsplit(data[i],/extract)
> if (line[0] eq 'i_shot_ctr') then shot = line[2:*]
> if (line[0] eq 'i_rng_wf' and n_elements(line) eq 4002) then $
> for j=0,n_elements(shot)-1 do printf, outunit,
> line[0]+'_'+shot[j]+' = ['+strjoin(line[j*200+2:j*200+201],', ')+']'
> ; This will not work. Max n_elements should be 251.
> if (line[0] eq 'i_rng_wf' and n_elements(line) eq 10882) then $
> for j=0,n_elements(shot)-1 do printf, outunit,
> line[0]+'_'+shot[j]+' = ['+strjoin(line[j*544+2:j*544+545],', ')+']'
> if (line[0] ne 'i_rng_wf') then printf, outunit,
> strjoin(line[0:1],' ')+' ['+ strjoin(line[2:*],', ')+']'
> endfor
> print, 'End of processing.'
> print, 'Type the following at the IDL prompt to display the
> variables:'
> print, "@"'+outfile+""
> close, outunit & free_lun, outunit
> end

This seems to be missing the procedure declaration, since it has an
end (batch files, which you would run with @, do not have the end

statement). When you run a procedure (or function, for that matter), all the variables defined inside them are in scope only until the end of that procedure. You do not see them in the variable view because they do not exist anymore once that procedure is done.

If you want some variables to be available after it finishes, add them as arguments to the procedure declaration, and use those arguments when calling it, to pass the variables back to the calling scope.

For instance, if your procedure is called read_some_variables, and you want to have the variables data and heads at the end, the procedure should be:

```
pro read_some_variables,data,heads
(...)
end
```

Then you call it as

```
read_some_variables,data,heads
```