Subject: Re: PRINT request for RSI
Posted by Martin Schultz on Tue, 03 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Robert S. Mallozzi wrote:
>
> Hi guys,
>
> IDL is a great tool for string manipulation, and I
> frequently use it to generate various files. However,
> it is quite painful to have to invoke the PRINT
> function for every line of output. I think it would
> be extremely useful for IDL to have the capability
> to use the unix shell or perl type of printing,
> where one specifies to print until a certain label.
> For example, in perl I can do
>
> print << EOF;
>
> print stuff
> print more stuff
>
> EOF
>
> As far as I can tell, there is no way to do something
> like this in IDL, is there? Maybe in IDL, something
> like this could be implemented:
>
> PRINT UNTIL LABEL
>
> 'x = ', x
> 'a string'
>
> LABEL:
>
> This, I think, would be a great feature.

Seems like you want to print the same thing all the time, because
you are coding it into your program. What I do in these cases is
write the text into an ASCII file, read this into a string array
and then print it. You can also use this approach to define "tokens"
that are replaced by actual values when output. I'll attach a lenghty
program that uses this method - perhaps you'll find it useful.

> On another topic, I would like to see some method
> implemented for commenting out whole blocks of
> code, rather than using ; on every line, or a

> GOTO statement. I can't think of any reason offhand
> what the C commenting contruct /* */ couldn't
> be used. Anyway, just my $0.02 (for today :-)
>
That's something I have wanted sometimes as well. It wouldn't
even have to be so fancy as to take care of nesting etc.

Martin

--- -------
Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
186 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
IDL-homepage: http://www-as.harvard.edu/people/staff/mgs/idl/
--- -------

```
;------------------------------------------------------------ --
;+
; NAME:
;       WRITEGTE
;
; PURPOSE:
;       Convert a data set (or data file) to NASA's GTE format.
;       Allows for flexible handling of header information.
;
; CATEGORY:
;       Data Input/Output
;
; CALLING SEQUENCE:
;       WRITEGTE,{(DATA,HEADER,UNITS)|FILENAME=FILENAME }[,keywords]
;
; INPUTS:
;       DATA, HEADER, UNITS --> data array, variable name list and unit
;       name list containing the data to be converted. If DATA contains
;       less than 2 elements, an input file must be provided with the
;       keyword parameter FILENAME.
;
; KEYWORD PARAMETERS:
;       FILENAME --> If no DATA is passed in the DATA parameter
;       (togetehr with the appropriate HEADER and UNITS information),
;       the data from this file will be read and converted into
;       GTE format. The file format is not entirely flexible: Although
;       the READDATA parameters SKP1, SKP2, AUTOSKIP and DELIM can
```

```
;       be used, WRITEGTE expects the UNITS declaration always in
;       the line immediately following the variable names (e.g. if
;       the variable names (HEADER) are in line 2 (i.e. SKP1=1), UNITS
;       must be in line 3).
;       After program execution, the parameters DATA, HEADER, and UNITS
;       contain the respective information of the data set read by FILENAME.
;
;       OUTFILE --> Name of the output file to be generated. Default is
;       'mydata.gte'.
;
;       SKP1, SKP2, AUTOSKIP, DELIM --> input parameters for the READDATA
;       routine which is called if no DATA is passed.
;
;       DESCFILE --> Name of a "descriptor" file which contains the
;       structure information of the GTE header *and* the comments section.
;       Default is 'gte_def.header'. This file is a simple ASCII file
;       which contains 'tokens' identified with leading '$$' (token
;       names must terminate with a blank or the end of a line). Each
;       token may be repeated several times but must be unique in it's
;       name and not contained in another token (e.g. $$VAR and $$VARIABLE
;       will both be replaced with the cotents of VAR). Otherwise, the user
;       may choose tokennames at will, and these will be replaced with
;       the tokenvalues supplied in TOKENVALUES or prompted interactively.
;       There are a few special tokens:
;         $$FILENAME = adds the filename of the output file without path
;            information
;         $$SYSDATE = adds the system date in the form YYYY/MM/DD
;         $$VARIABLE_NAMES = generates a comma seperated list of variable
;            names (i.e. the entries of the HEADER field)
;         $$VARIABLE_DEFINITIONS = generates the GTE variable definition
;            block with NAME, UNIT, SCALE (always 1), OFFSET (always 0),
;            MINIMUM, MAXIMUM, MISSING VALUE, DL-coding   for each variable.
;         $$NVARS = inserts the number of variables in the data set
;         $$NCOMMENTS = inserts the number of comment lines in the data set
;         $$NHEADERLINES = inserts the total number of header lines
;
;       TOKENNAMES --> A string array with tokennames that shall be set
;       to the corresponding values of TOKENVALUES. All tokens found in the
;       description file that are neither special tokens nor passed in
;       TOKENNAMES will be prompted interactively.
;
;       TOKENVALUES --> A string array containing the values for the tokens
;       passed in TOKENNAMES.
;
;       /TEST --> set this flag to test the formatting of the header. Only
;       5 lines of data will be written.
;
;
```

```
; OUTPUTS:
;       A data file that adheres to the NASA GTE specification (if the description
;       file is correct).
;
; SUBROUTINES:
;       pro add_variabledef,text,line,data,header,units
;            constructs the variable definition block and inserts it into the
;            current header
;
;       pro replace_token,text,token,tokenresult,data,header,units
;            replace the $$TOKENNAME in the header description with it's value.
;            NOTE: One token can only be replaced once in each line of the header.
;
;       function find_tokens,text
;            compiles a list of all tokens in the description file.
;
; REQUIREMENTS:
;       Uses READDATA and requires a header description file
;
; NOTES:
;
; EXAMPLE:
;       WRITEGTE,file='this.data',skp1=1,skp2=1,delim=' ',  $
;              out='thisdata.gte',desc='gtepemt.header',  $
;              tokennames=['MISSIONNAME','MISSIONDATE',   $
;              'AIRCRAFT'],  $
;              tokenvalues=['PEM-Tropics','1996/08/30',   $
;              'DC-8' ]
;
;       produces a GTE conform data set of PEM-Tropics data from the file
;       this.data .
;
; MODIFICATION HISTORY:
;       mgs, 05 Jan 1998: VERSION 1.00
;
;-
; Copyright (C) 1998, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine writegte"
 ;--------------------------------------------------------- --
```

```
pro add_variabledef,text,line,data,header,units


    nv = n_elements(header)
    vdef = strarr(nv)

    ; fake cfact -- should be 1.0 anyway
    cfact = fltarr(nv) + 1.

    for i=0,nv-1 do begin
      x = data(i,*)
      ind=where(x gt -666,c)
      if (c gt 0) then x = x(ind)
      vdef(i) =  "" +header(i)+ "","" +units(i)+ "',' + $
            string(cfact(i),format='(f6.1)')+    $
            ', 0.0, ' +  $
            string(min(x))+ $
            ', ' +  $
            string(max(x))+ $
            ', -999.99, 0'
      if(max(x) le -666) then print,header(i),' ONLY MISSING VALUES !!'
      if(units(i) eq 'X' OR units(i) eq 'unknown') then $
         print,header(i),' :  units = ',units(i)
    endfor

    nlines = n_elements(text)

    tmptext = [ text(0:line-1), vdef, text(line+1:nlines-1) ]
    text = tmptext

return
end



pro replace_token,text,token,tokenresult,data,header,units

; limitation: each token can only be present once in each line
; (you can however have several tokens in one line, this procedure is
; called for every token !)
; while instead of for loop because number of lines can change

; common block holds number of lines after variable definition block
; (i.e. number of comment lines)
common nlines,ncomment

   if (n_elements(ncomment) le 0) then ncomment = -1
```

```
   i = 0
   while (i lt n_elements(text)) do begin
      p = strpos(text(i),'$$'+token)
      if (p ge 0) then begin
         if (strmid(tokenresult,0,1) eq '$') then begin  ; special tokens
         ; $1 : variable definition block
            if (tokenresult eq '$1') then begin
               ncomment = n_elements(text) - i - 1
               add_variabledef,text,i,data,header,units
               tokenresult = ''
            endif

         ; $2 : number of comment lines
            if (tokenresult eq '$2') then $
               tokenresult = string(ncomment,format='(i0)')

         ; $3 : total number of header lines
            if (tokenresult eq '$3') then $
               tokenresult = string(n_elements(text),format='(i0)')
         endif

         if (tokenresult ne '') then begin
         ; normal tokens (replacement within line)
         ; split string into part before and after token
         s1 = strmid(text(i),0,p-1)
         p2 = strpos(text(i),' ',p+2)   ; position of first blank after token
         if (p2 ge 0) then s2 = " "+strmid(text(i),p2+1,256) else s2 = ""
         ; re-compose text line
         text(i) = s1 + tokenresult + s2
         endif
      endif
      i = i+1
   endwhile

return
end



function find_tokens,text

   tokens = ''
   for i=0,n_elements(text)-1 do begin
      p = 0
      while (p ge 0) do begin
         p = strpos(text(i),'$$',p)
```

```
        if (p ge 0) then begin
          newtoken = str_sep(strupcase(strmid(text(i),p+2,25)),' ')
          if (newtoken(0) ne '') then $
             tokens = [ tokens, newtoken(0) ]
          p = p+2
        endif
      endwhile
    endfor


    if (n_elements(tokens) gt 1) then tokens = tokens(1:n_elements(tokens)-1)

return,tokens
end




pro  writegte,data,header,units,filename=filename,outfile=outfile , $
      descfile=descfile, $
      delim=delim,skp1=skp1,skp2=skp2,autoskip=autoskip, $
      tokennames=tokennames,tokenvalues=tokenvalues, $
      test=test


on_error,2

    ; default parameters
    if (not keyword_set(outfile)) then outfile = 'mydata.gte'
    if (not keyword_set(descfile)) then descfile = 'gte_def.header'

    ; check if tokenvalues were passed and add default values.
    ; Since default values will be appended and only the first entry
    ; for each token is used, defaults are overwritten by command line
    ; passed arguments
    if (n_elements(tokennames) le 0) then begin
       tokennames = '***'
       tokenvalues = '***'
    endif
    if (n_elements(tokennames) ne n_elements(tokenvalues)) then $
       message,"Number of tokennames <> number of tokenvalues !"

    tokennames = [ strupcase(tokennames), $
           "INVESTIGATOR", "PRODUCT", "MISSIONNAME", $
           "MISSIONDATE", "FLIGHTNUMBER", "DATASETTYPE", $
           "AVERAGING_TIME", "SAMPLING_FREQ"  ]

    tokenvalues = [ tokenvalues, $
```

```
        "Daniel J. Jacob and Martin G. Schultz", $
        "Harvard University point model calculation", $
        "PEM-Tropics (A)", $
        '1996/08/30', $
        '0', '0', '0.', '0.' ]



; check if data has been passed or must be read
if (n_elements(data) lt 2 OR n_elements(header) lt 1 OR $
    n_elements(units) lt 1) then begin
    if (not keyword_set(filename)) then $
       message,"WRITEGTE: requires either input data or filename !"

    if (n_elements(delim) le 0) then delim = ' '
    if (n_elements(skp1) le 0) then skp1=1
    if (n_elements(skp2) le 0) then skp2=1

     readdata,filename,data,header,comments=comments,skp1=skp1,sk p2=skp2, $
           delim=delim,autoskip=autoskip

    units = extract_comments(comments,skp1+1)   ; units must be in line after
                                        ; variable names
    ; hope this crashes if attempt is made to read file without unit definitions

    ; here should be a chance to change the title line, for now we print it only
    if (n_elements(comments) ge 2) then title = comments(0) else title=''
    print,'Title line extracted from file '+filename
    print,title
endif



; read description file with comment lines
openr,dlun,descfile,/get_lun
char = ''
text = ''
while (not eof(dlun)) do begin
   readf,dlun,char
   text = [ text, char ]
endwhile
close,dlun
free_lun,dlun

if (n_elements(text) gt 1) then text = text(1:n_elements(text)-1) $
else print,"No tokens in description file !!"



; extract tokens and try to fill them with values
```

```
tokens = find_tokens(text)
print,'Tokens found in description file : '
for i=0,n_elements(tokens)-1 do print,tokens(i)
print

tokenresults = strarr(n_elements(tokens))
for i=0,n_elements(tokens)-1 do begin
   ; first the easy ones
   if (tokens(i) eq 'FILENAME') then $
      tokenresults(i) = extract_filename(outfile)

   if (tokens(i) eq 'NVARS') then $
      tokenresults(i) = string(n_elements(header),format='(i0)')

   if (tokens(i) eq 'SYSDATE') then $
      tokenresults(i) = strdate(/short,/sortable)

   if (tokens(i) eq 'VARIABLE_NAMES') then $
      tokenresults(i) = string(header,format='(512(A,:,","))')

   ; use another level of templates for the more complicated ones
   ; the number determines the order in which they will be processed
   if (tokens(i) eq 'VARIABLE_DEFINITIONS') then $
      tokenresults(i) = "$1"

   if (tokens(i) eq 'NCOMMENTS') then $
      tokenresults(i) = "$2"

   if (tokens(i) eq 'NHEADERLINES') then $
      tokenresults(i) = "$3"

   ; all others: check if entry exists in tokennames, else query
   if (tokenresults(i) eq "") then begin
      j = where(tokennames eq tokens(i))
      if (j(0) ge 0) then tokenresults(i) = tokenvalues(j) $
      else begin
         char = ""
         read,char,prompt="Enter value for token "+tokens(i)+" : "
         tokenresults(i) = char
         tokennames = [ tokennames, tokens(i) ]
         tokenvalues = [ tokenvalues, char ]
      endelse
   endif
endfor


; now process token replacement into text array
; start with special tokens, then all the rest
```

```
    ind = where(tokenresults eq '$1')
    if (ind(0) ge 0) then $
          replace_token,text,tokens(ind(0)),tokenresults(ind(0)),data, header,units

    ind = where(tokenresults eq '$2')
    if (ind(0) ge 0) then $
        replace_token,text,tokens(ind(0)),tokenresults(ind(0))

    ind = where(tokenresults eq '$3')
    if (ind(0) ge 0) then $
        replace_token,text,tokens(ind(0)),tokenresults(ind(0))

    for i=0,n_elements(tokens)-1 do $
        replace_token,text,tokens(i),tokenresults(i)




    ; now we can write the new file
    print,'writing to file '+outfile
    openw,unit1,outfile,width=4096,/get_lun

    ; find index of first photolysis frequency
    for i=0,n_elements(header)-1 do begin
      p=strpos(strupcase(header(i)),'J')
      if (p ge 0) then begin
        ifrc=i
        goto,frcfound
      endif
    endfor
    print,' *** INDEX OF FIRST PHOTLOYSIS FREQ. NOT FOUND !!! ***'
    stop

frcfound:


     for i=0,n_elements(text)-1 do $
        printf,unit1,text(i)


     dummy = data(0,*)

     ndat = n_elements(dummy)
     if (keyword_set(test)) then ndat = 5

     for i=0,ndat-1 do begin
       dataline = data(*,i)
       dats = string(dataline(0),dataline(1),dataline(2),   $
```

```
            format="(i4,2(',',f9.1))")
  for j=3,n_elements(header)-1 do  $
     if (header(j) eq 'mDENSITY' OR (j ge ifrc)) then  $
          dats = dats+','+string(dataline(j),format='(e11.4)')  $
     else dats = dats+','+string(dataline(j),format='(f10.2)')
   printf,unit1,dats
endfor



close,unit1

free_lun,unit1

print,' DONE.'


return
end
```

File Attachments

1) writegte.pro, downloaded 98 times