MWSUG 2016 — Paper TT06

List Processing Macro Function CallText: Read a Control Data Set and Unquote each Item

Ronald J. Fehd, Stakana Analytics

Abstract

Description: The macro language of SAS® software provides the %nrstr function which can be used to delay resolution of macro variable references until execution time, at which time the %unquote function is used to call for the resolution of reference. This paper shows the issues in writing a macro function which reads a list, a control data set, and returns each item surrounded by user-provided special characters. This macro uses `scl` functions to read the data set thus avoiding the necessity of a data step.

Purpose: The purpose of this paper is to provide a working example of a macro function which calls explicitly state the special characters desired to enclose each item in the list.

Audience: programmers writing list processing applications

Keywords: hexadecimal, functions: `datetime, getoption, putn`;
macro functions: `%nrstr, %sysfunc, %unquote`;
`scl` functions: `attrn, close, exist, fetchobs, getvarc, getvarn, open, varname, varnum, vartype`

In this paper

Introduction

Overview

The section contains these topics.

- description
- parameters
- algorithm

description

Macro `calltext.sas` is a list-processing function and routine which reads a control data set, converts each variable and its value in each observation into macro variables and then unquotes the parameter `text` which contains macro variable references as well as quotes and other special characters.

parameters

This is the list of parameters.

- `data`, one- or two-level data set name
- `text`, text with macro variable references and special characters
- `delimiter`, text to add between expansion of parameter `text`
- `global`, boolean: write macro variables to global symbol table; i.e.: make macro variables from row available to other processes
- `hex16`, boolean: convert numerics to hexadecimal

algorithm

This is the list of steps in the algorithm.

1. assertion: data exists
2. fetch data information: n-obs and n-vars
3. assertion: data has n-obs and n-vars
4. for each observation:
   (a) for each variable:
      i. fetch variable information: name, number and type
      ii. allocate macro variable
   (b) return the string contained in the parameter `text`
   with macro variable references expanded

hard-coded example

This program shows the desired results of the function.

```
*function returns tokens within DATA statement;
DATA work.class_sex_F
     work.class_sex_M
     ;
do until(endofile);
   set sashelp.class end = endofile;
   *returns statements and delimiter;
   if sex eq     'F' then output
   work.class_sex_F;
   %*delimiter; else
   if sex eq     'M' then output
   work.class_sex_M;
   end;
stop;
run;
```

Programs, Log and Listing

Overview

This is the list of programs, logs and listings shown here.

- macro calltext.sas
- program calltext-test.sas
- subroutine proc print with titles
- calltext-test.log
- calltext-test.lst

macro calltext

```
 /* name: <UNC>\SAS-site\macros\calltext.sas
  author: Ronald J. Fehd 2012
          --------------------------------------------------
Summary : description  : call text with specified
                         values in data set as parameters
          purpose      : provide generic method to call text
                         using list==control data set of parms
          --------------------------------------------------
Contexts: program group: list processing token generator
          program  type: function
          SAS      type: macro assignment statement generator
          uses routines: n/a
          --------------------------------------------------
Specifications: input  : required: data, text
                         optional: delimiter, hex16
                process: assemble macro variable assignment(s)
                         call
                output : from text
Parameters: data    = one- or two-level data set name
           ,text    = text of macro references for each row
      *-constraint-*: must be enclosed in nrstr:
           ,text    = %nrstr(%put note: name:&name sex=&sex;)
                      notes: may be any of:
                      * tokens
                      * statements including semicolons
                      * macro calls:
           ,text    = %nrstr(%mymacro(name=&name,sex=&sex))
           ,hex16   = 1 :: default, convert numeric to hex16
                      used to pass real numbers accurately
                           across step boundaries
           ,hex16   = 0 :: pass numerics as decimals
           ,testing = 0 :: default, no extra messages in log
           ,testing = 1 :: note: auto-reset when
                           options mprint source2;
           ,unquote = 1 :: default, write notes to log
           ,unquote = 0 :: do not   write notes to log
            --------------------------------------------------
bells, whistles: writes real time used to log
                 note: CALLTEXT used real time  0:00:00.016
-----------------------------------------------------------
```

```
usage example:
PROC freq data   = sashelp.class;
          tables   sex / noprint
             out = freq_class_sex;
run;*necessary;
DATA %calltext(data = freq_class_sex
              ,text = %nrstr(work.sex_&sex)
              );*end of data statement;
do until(endofile);
   set sashelp.class  end = endofile;
   %calltext(data      = freq_class_sex
            ,delimiter = %nrstr(else)
            ,text
   =%nrstr(if sex eq "&sex" then output work.sex_&sex;)
            )
   end;
stop;
run;
log: ---------------------------------------------------
13     DATA
MPRINT(CALLTEXT):   work.sex_F
MPRINT(CALLTEXT):   work.sex_M
18     do until(EndoFile);
19        set sashelp.class end = endoFile;
MPRINT(CALLTEXT): if Sex eq "F" then output work.sex_F;
MPRINT(CALLTEXT): else
MPRINT(CALLTEXT): if Sex eq "M" then output work.sex_M;
25        end;
26     stop;
27     run;
NOTE: There were 19 obs read from data set SASHELP.CLASS
NOTE: The data set WORK.sex_F has  9 obs and 5 variables
NOTE: The data set WORK.sex_M has 10 obs and 5 variables
---------------------------------------------------------
DateTime: 2016-09-09 7:10:13 AM
word count:       words:  691
                  lines:  157
characters(no   spaces): 4591
characters(with spaces): 6584
****** ..................... */
%MACRO calltext
        (data      = sashelp.class
        ,text      = %nrstr(%put note: name:&name sex=&sex;)
        ,delimiter= /* %nrstr(else) */
        ,global    = 0 /* make mvars global? */
        ,hex16     = 1 /* convert numerics to hex16? */
        ,testing   = 0
        ,unquote   = 1 /* write notes to log? */
)/ des =
'site: assemble mvar assignment statement(s), unquote text'
 /** * store source /* */
;/**** NOTE: _*: avoid name collisions w/data set vars ***/
%local _delimiter _global _hex16 _text _testing _unquote
       _dsid _rc  _obs _n_obs    _var  _n_vars
       _name _num _type          _time_start _time_end;
%let _testing = %eval(not(0 eq &testing)
        or(    %sysfunc(getoption(mprint))  eq MPRINT
           and %sysfunc(getoption(source2)) eq SOURCE2));
%let _time_start= %sysfunc(datetime(),hex16);
%let _delimiter = &delimiter;
%let _global    = %eval(not(0 eq &global));
%let _hex16     = %eval(not(0 eq &hex16));
%let _text      = &text;
%let _unquote   = %eval(not(0 eq &unquote));

%**  description: assertions;
%**  purpose    : if fail then exit;
%if  not(%sysfunc(exist(&data))) %then %do;
     %put Err%str()or &sysmacroname: not exist(&data);
     %return;
     %end;
%let _dsid  = %sysfunc(open (&data         ));
%let _n_obs  = %sysfunc(attrn(&_dsid,nobs ));
%let _n_vars = %sysfunc(attrn(&_dsid,nvars));
%put info: &sysmacroname &=data &=_n_obs &=_n_vars;
%if  not &_n_obs or not &_n_vars %then %do;
     %put Err%str()or &sysmacroname exit: not(nobs and vars);
     %goto close_exit;
     %end;

%** description: read data;
%** purpose    : make macro variable assignment(s), submit;
%do _obs = 1 %to &_n_obs;%*** fetchobs==read row;
    %let _rc       = %sysfunc(fetchobs(&_dsid,&_obs ));
    %do _var = 1 %to &_n_vars;
        %let _name = %sysfunc(varname (&_dsid,&_var ));
        %let _num  = %sysfunc(varnum  (&_dsid,&_name));
        %let _type = %sysfunc(vartype (&_dsid,&_num ));
        %if &_global %then %global &_name;
        %** assign: for type=C: value, type=N: hex16(value);
        %if   &_type eq C
           or(&_type eq N and not &_hex16) %then
              %let &_name = %left(%sysfunc(
                       getvar&_type(&_dsid,&_num)));
        %else %let &_name = %left(%sysfunc(putn(
              %sysfunc(getvar&_type(&_dsid,&_num)),hex16)));
        %if &_testing or &_global %then
            %put echo: &_name=&&&_name;
        %end;

    %if not &_testing and &_unquote %then
        %put &sysmacroname output: %cmpres(%unquote(&_text));
    %*** submit: note! no ending semicolon!;
    %unquote(&_text)

    %if %length(&_delimiter) and &_obs lt &_n_obs %then %do;
        %if not &_testing and &_unquote %then
            %put &sysmacroname output: %unquote(&_delimiter);
        %unquote(&_delimiter)
        %end;
    %end;
%if &_testing %then %put _local_;
```

```

%close_exit: %let _rc       = %sysfunc(close(&_dsid));
             %let _time_end = %sysfunc(datetime(),hex16);
%put info: &sysmacroname used real time %sysfunc
           (putn(&_time_end.x-&_time_start.x,time12.3));
%mend calltext;
```

program calltext-test.sas

```
* name: call-text-test-class-print.sas;
options mprint;* source2; * testing on;
%let name = sex;
PROC freq data   = sashelp.class;
          tables   &name / noprint
             out = freq_class_&name;
run;*necessary;
DATA %calltext(data = freq_class_&name
              ,text = %nrstr(work.&name._&&&name)
              );*end of data statement;
do until(endofile);
   set sashelp.class end = endofile;
   %calltext(data      = freq_class_&name
            ,delimiter = %nrstr(else)
            ,text=%nrstr
  (if &name eq "&&&name" then output work.&name._&&&name;)
            )
   end;
stop;
run;
options nomprint nosource2; * testing off;
*replicates method of CallXinc::cxinclude;
%let sex    =.;
%let count  =.;
%let percent=.;
%calltext(data     = freq_class_&name
         ,text     = %nrstr
         (%include project(proc-print-subset-w-titles);)
        ,global    = 1)
%put &=sex &=count &=percent;
```

log

```
8     DATA %calltext(data = freq_class_sex
9                   ,text = %nrstr(work.sex_&sex)
10                  );*end of data statement;
MPRINT(CALLTEXT):   work.sex_F
MPRINT(CALLTEXT):   work.sex_M
11    do until(endofile);
12       set sashelp.class  end = endofile;
13       %calltext(data      = freq_class_sex
14                ,delimiter = %nrstr(else)
15                ,text
16       =%nrstr(if sex eq "&sex" then output work.sex_&sex;)
17                )
MPRINT(CALLTEXT): if sex eq "F" then output work.sex_F;
MPRINT(CALLTEXT): else
MPRINT(CALLTEXT): if sex eq "M" then output work.sex_M;
18            end;
19        stop;
20        run;
NOTE: There were 19 obs read from the data set SASHELP.CLASS.
NOTE: The data set WORK.SEX_F has  9 observations and 5 variables
NOTE: The data set WORK.SEX_M has 10 observations and 5 variables
info: CALLTEXT DATA=freq_class_sex _N_OBS=2 _N_VARS=3
CALLTEXT output: %include project(proc-print-subset-w-titles);
echo parameters NAME=sex: F COUNT  =4022000000000000
                            PERCENT=4047AF286BCA1AE7
```

routine proc print with titles

This routine shows how to format macro variables saved as hex16 back to integer and real numbers.

```
*name: proc-print-subset-w-titles.sas;
*description: shows how to convert numbers in mvars saved as hex16
              into integers and reals;
%put echo parameters &=name: &&&name &=count &=percent;
PROC print data   = work.&name._&&&name;
           Title2 "data    : work.&name._&&&name";
           Title3 "subset  : &name eq &&&name";
           Title4 "freq    : count  : &count";
           Title5 "        : percent: &percent";
           Title6 "   count: %left(%sysfunc(putn(&count.x  ,32.  )))";
           Title7 " percent: %left(%sysfunc(putn(&percent.x,32.16)))";
           Title8 " rounded: %left(%sysfunc(putn(&percent.x,6.2  )))";
run;
```

partial listing

```
List Processing macro function CallText
data    : work.sex_F
subset  : sex eq F
freq    : count  : 4022000000000000
        : percent: 4047AF286BCA1AE7
   count: 9
 percent: 47.0000000000000000
 rounded: 47.00

Obs    Name       Sex    Age    Height    Weight
---    -------    ---    ---    ------    ------
 1     Alice       F      13     56.5       84.0
 2     Barbara     F      13     65.3       98.0
```

Summary

Suggested Reading

This function is a sibling of list processing macro function call-macro which is described in Fehd [5]. Both `call-macro` and `call-text` are derivative works of the list processing routine CallXinc described in Fehd [4], which is a derivative work of the SmryEachVar suite Fehd [3].

list processing : Fehd and Carpenter [6] provides an overview of the issues of creating and processing lists of macro variables

macro processor : Lavery [7] provides a definitive overview of the interaction of the tokenization of macro and SAS statements; Chaudhary [1] discusses differences between macro compile and execution functions; Russell [11], author from SAS Tech Support provides answers to top 10 FAQ (Frequently Asked Questions); Lee [8] compares macro functions `%nrstr` and `%nrbquote`; Rosenbloom and Carpenter [10] shows how to store and retrieve special characters in macro variables;

macros for lists : Clay [2] discusses vertical arrays of macro variables and provides examples of macro functions; Tsai [12] discusses use of horizontal and vertical arrays of macro variables; Whitlock [13], an acknowledged master of macro usage, discusses use of quotes and separators when writing macro functions which process lists of tokens; Morris [9] shows common problems in macros for list processing

Conclusion

SAS Component Language (SCL) functions can be used to open a data set, read each row, and, within each row, fetch variable information which is used to allocate macro variables; with a list of macro variables for each row the parameter `text` can then be expanded returning strings which may be tokens within a SAS statement or one or more statements. The use of the pair of macro functions, `%nrstr` during compilation and `%unquote` during execution, allows a programmer to write code which is easily understood.

Author Information

Ronald J. Fehd `Ron.Fehd.macro.maven@gmail.com`

About the author:

sco.wiki	`http://www.sascommunity.org/wiki/Ronald_J._Fehd`
LinkedIn	`www.linkedin.com/Ronald.Fehd`
affiliation	Stakana Analytics, Senior Maverick
also known as	`macro maven` on SAS-L, Theoretical Programmer

Programs: `http://www.sascommunity.org/wiki/` Macro_CallText

Trademarks

References

[1] Kaushal Raj Chaudhary. Essentials of macro quoting functions in SAS(R). In *MidWest SAS Users Group Annual Conference Proceedings*, 2015. URL `http://www.mwsug.org/proceedings/2015/RF/MWSUG-2015-RF-08.pdf`. Rapid Fire, 4 pp., introduction to macro quoting functions, discusses difference between compile (assignment) and execution (resolution) functions.

[2] Ted Clay. Tight looping with macro arrays. In *SAS Users Group International Annual Conference Proceedings*, 2006. URL `http://www2.sas.com/proceedings/sugi31/040-31.pdf`. 8 pp., covers vertical macro arrays, shows many examples of list processing and enclosure of items in list.

[3] Ronald J. Fehd. SmryEachVar: A data-review routine for all data sets in a libref. In *SAS Global Forum Annual Conference Proceedings*, 2008. URL `http://www2.sas.com/proceedings/forum2008/003-2008.pdf`. Applications Development, 24 pp.; call execute, data review, data structure, dynamic programming, list processing, parameterized includes, utilities (writattr, writvalu) to repair missing elements in data structure; best contributed paper in ApDev.

[4] Ronald J. Fehd. List processing routine CallXinc: Calling parameterized include programs using a data set as list of parameters. In *Western Users of SAS Software Annual Conference Proceedings*, 2009. URL `www.lexjansen.com/wuss/2009/app/APP-Fehd2.pdf`. Applications Development, 20 pp.; call execute, data review, data structure, dynamic programming, list processing, parameterized includes, examples.

[5] Ronald J. Fehd. List processing macro call-macro. In *MidWest SAS Users Group Annual Conference Proceedings*, 2014. URL `www.mwsug.org/proceedings/2014/BB/MWSUG-2014-BB04.pdf`. Coders Corner, 19 pp.; using %sysfunc with SCL functions to read a list, a control data set, and for each observation, call a macro with variable names and values as named parameters.

[6] Ronald J. Fehd and Art Carpenter. List processing basics: Creating and using lists of macro variables. In *SAS Global Forum Annual Conference Proceedings*, 2007. URL `http://www2.sas.com/proceedings/forum2007/113-2007.pdf`. Hands On Workshop, 20 pp.; comparison of methods: making and iterating macro arrays, scanning macro variable, writing calls to macro variable, write to file then include, call execute; using macro function nrstr in call execute argument; 11 examples, bibliography.

[7] Russ Lavery. An Animated Guide(C): How %str and %nrstr work. In *NorthEast SAS Users Group Conference Proceedings*, 2004. URL `http://www.lexjansen.com/nesug/nesug04/pm/pm25.pdf`. Programming and Manipulation, 13 pp., thorough explanation of the difference between six types of compile and execution of macro tokens before SAS tokens; discusses functions used to mask special characters; recommended.

[8] Shan Lee. Quoting macro variable references. In *Pharmaceutical Users of Software Exchange*, 2007. URL `http://www.lexjansen.com/phuse/2007/tu/TU04.pdf`. Tutorials, 6 pp., comparison of functions `%nrstr` and `%nrbquote` with single and double quotes.

[9] Robert J. Morris. Text utility macros for manipulating lists of variable names. In *SAS Users Group International Annual Conference Proceedings*, 2005. URL `http://www2.sas.com/proceedings/sugi30/029-30.pdf`. Coders Corner, 7 pp., examples of common problems of quoting items in a list.

[10] Mary F. O. Rosenbloom and Art Carpenter. Macro quoting to the rescue: Passing special characters. In *SAS Global Forum Annual Conference Proceedings*, 2013. URL `http://support.sas.com/resources/papers/proceedings13/005-2013.pdf`. Applications Development, 12 pp., storing and retrieving special characters; using macro quoting functions.

[11] Kevin Russell. SAS(R) macros: Top ten questions (and answers!). In *MidWest SAS Users Group Annual Conference Proceedings*, 2010. URL `http://www.mwsug.org/proceedings/2010/advanced/MWSUG-2010-174.pdf`. section 39 slides; `%sysfunc` with data step functions, quoting functions to mask special characters, resolving macro variable references inside single quotes.

[12] Jerry S. Tsai. Efficiently handle lists of items using the SAS Macro Language — save time and reduce errors. In *Western Users of SAS Software Annual Conference Proceedings*, 2008. URL `http://www.lexjansen.com/wuss/2008/cod/cod06.pdf`. Coders Corner, 8 pp., use of horizontal and vertical arrays of macro variables in utility macros; best contributed paper.

[13] Ian Whitlock. Names, names, names — make me a list. In *SAS Global Forum Annual Conference Proceedings*, 2007. URL `http://www2.sas.com/proceedings/forum2007/052-2007.pdf`. Coders Corner, 10 pp.; binary list operators, use of double quotes and delimiters/separators, uses of macro functions `%bquote`, `%qscan`, `%str`, `%superq`, `%unquote`.