Paper 81-2010

The Genealogy of Macros

Kelley Weston; Quintiles; Overland Park, KS

Abstract

When one SAS® macro calls another macro, which calls another macro (etc., ad infinitum), how can you find what the "macro genealogy" is? This paper will give an easy method that you can use to find all of the macros involved in your program. This method is SAS-version independent, and will work on virtually any operating system.

Keywords

OPTIONS MLOGIC

Introduction

Why would you want to find out all of the macros involved in running your program?

How can you find them all?

The answer to the first is two-fold – debugging your program, when there are many macros involved, you may need to track down other macros (that may or may not be causing you problems); and documenting your program (so all of the necessary macros are known and are available).

The answer to the second is the focus of this paper. We will make use of the SAS system option MLOGIC, and the SAS log.

Tracing the flow of the program

With the SAS system mlogic, you see lines in the log similar to this:

```
MLOGIC (my_macro): Beginning execution.

MLOGIC (my_macro): Ending execution.
```

We can take advantage of this, using the log, not just to read to see if we had any errors or warnings, but as an input file to our program to trace the macro calls.

We want a simple report that will list the macros involved in our program, and will also indicate whether one macro is calling another macro, and to what level of nesting.

Let’s say that we want a report that looks similar to this:

```
                    Trace for My_macro

                     Obs Level & Macro

                       1  01  My_macro
                       2  02    Macro01
                       3  03      Macro02
                       4  02    Macro03
                       5  03      Macro04
                       6  04        Macro05
```

This indicates that **my_macro** directly calls the level 02 macros **macro01** and **macro03**; the level 02 macros directly call the level 03 macros – **macro01** calls **macro02**, and **macro03** calls **macro04**; and finally, the level 03 macro **macro04** calls **macro05**.

With the level numbers and the style of indentation, you can easily see all macros involved in the execution of the program, and their relation to each other (hence my term “genealogy” of macros).

Since this program creates a dataset with the macro names, we can easily add another useful functionality – an alphabetical list of the macros that are called, along with the number of times that each macro is called.

For instance, we could produce something like the following:

```
          Alphabetical List of Macros for My_macro

                    Macro      # of Macro
               Obs  Name       Occurrences

                1   Macro01         1
                2   Macro02         3
                3   Macro03         7
                4   Macro04         2
```

One other useful thing to include in this analysis program is the option to include or exclude SAS-supplied macros, such as %LEFT or %TRIM, since they may potentially be called many times.

Since we have the number of times that each macro is called, we could then further analyze the program to see if any macro could be made more efficient, thus perhaps saving execution time, disk space, etc. If so, we might want to start with the macros that execute the most number of times. This list gives us a good starting point.

Conclusion

There are occasionally needs for us as programmers to analyze our programs. We may look through them to see if we can make them any more efficient, but this is an unusual method of analysis in that we are analyzing the log to get an overview of the program.

References

SAS Online Documentation, SAS System Options

About the Author

Kelley has over 18 years experience with SAS programming. Kelley has been a programmer with a major telecommunications company and a SAS instructor teaching SAS classes nationwide, as well as teaching hands-on workshops at regional SAS conferences and PharmaSUG. His programming experience also includes clinical trials with a major pharmaceutical company and an international CRO. He is an active member of KCASUG (Kansas City Area SAS Users Group), including stints as Webmaster (2006 & 2007) and Chairman (since 2008).

Contact Information

Your comments and questions are valued and encouraged. Contact the author at the following:

Kelley Weston
c/o Quintiles
6700 W. 115th St.
Overland Park, KS 66211
Kw.mwsug@gmail.com

Trademarks

Appendix

Program that creates the above reports

```
%MACRO trace(file    =        /* name of log file with or without .log extension */
           , ext     = log    /* file extension of log file                      */
           , sh_lg   = short  /* short or long listing                           */
                              /* short: omits SAS-supplied macro calls           */
                              /* long:  includes SAS-supplied macro calls        */
           , alpha   = Y      /* whether or not to produce a second listing that */
                              /* is an alphabetical listing of the macros        */
                              /* and a # of their occurrences                    */
           , logfile =        /* path of log file                                */
           );
   %LOCAL logfile dsid nobs rc;
   %IF %INDEX(&file, .log) NE 0
       %THEN %LET file = %TRIM(%LEFT(%SCAN(&file, 1, %str(.))));

   %IF %LENGTH(&logfile) EQ 0 %THEN %DO;
        %PUT Name of log file must be supplied;
        %GOTO finish;
   %END;

   FILENAME trace "&logfile";

   %* check existence of log file;
   %IF NOT %SYSFUNC(FEXIST(trace))
       %THEN %GOTO nofile;

   %LET alpha = %UPCASE(%SUBSTR(%TRIM(%LEFT(&alpha))), 1, 1);
   %IF &alpha NE Y and
       &alpha NE N
       %THEN %LET alpha = Y;

   %* if sh_lg NE short or long, then default to short;
   %IF %INDEX(short/long, %LOWCASE(&sh_lg)) EQ 0 %THEN %DO;
         %LET sh_lg = short;
         %PUT Using Short listing (omitting SAS-Supplied macro calls);
   %END;

   DATA work._temp;
     INFILE trace TRUNCOVER;
     INPUT @01 line $char200.;
     %* subset for desired lines;
     IF ((INDEX(line, 'Beginning') GT 0) OR (index(line, 'Ending') GT 0)) AND
         (INDEX(line, 'execution') GT 0);
   RUN;

   DATA work._temp (KEEP = macro macro2 );
     LENGTH indent $10;
     RETAIN indent ' '
            cnt    0;
     SET work._temp END = end;

     line2  = SUBSTR(line, index(line, '('));
     macro  = SCAN(line2, 1);
     b_e    = MAX(indexw(line2, 'Beginning'), INDEXW(line2, 'Ending'));
     line3  = SUBSTR(line2, b_e);
     indent = SCAN(line3, 1);


     IF (NOT end)
        THEN DO;
           IF indent EQ 'Beginning' THEN DO;
              macro2 = PUT(cnt + 1, z3.) !! REPEAT(' ', cnt * 2) !! macro;
              cnt + 1;
           END;
           ELSE DO; * indent = 'Ending';
             macro2 = PUT(cnt + 1, z3.) !! REPEAT(' ', cnt * 2) !! macro;
             cnt + (-1);
             if cnt LT 0
                THEN cnt = 1;
           END;
        END;
        ELSE DO;  * print just the 1st macro call;
             macro2 = PUT(1, z3.) !! ' ' !! macro;
     END;

     %IF &sh_lg EQ short %THEN %DO;
          IF macro IN ('LEFT', 'TRIM', 'CMPRES', 'LOWCASE', 'QCMPRES', 'QLEFT',
                       'QLOWCASE', 'QTRIM', 'VERIFY')
             THEN DELETE;
     %END;

     IF (indent EQ 'Beginning') OR
        end
        THEN OUTPUT;
   RUN;

   %LET dsid = %SYSFUNC(OPEN(_temp));
   %LET nobs = %SYSFUNC(ATTRN(&dsid, nobs));
   %LET rc   = %SYSFUNC(CLOSE(&dsid));

   %IF &nobs GT 0
       %THEN %DO;
       TITLE " Trace for &file";
       PROC PRINT DATA    = work._temp
                  HEADING = H
                  N
                  L;
         LABEL macro2 = 'Level & Macro';
         VAR macro2;
       RUN;
       TITLE;
       %IF &alpha EQ Y
          %THEN %DO;
            PROC SORT DATA = _temp;
                  BY macro;
            RUN;

            DATA work._temp;
            SET work._temp;
             BY macro;
             IF first.macro
                    THEN cnt = 0;
              cnt + 1;
              IF LAST.macro
                  THEN OUTPUT;
             RUN;

             TITLE " Alphabetical list of macro calls for program &file";
             PROC PRINT DATA    = work._temp
                       HEADING = h
                       N
                       L;
              LABEL macro = 'Macro Name'
                    cnt   = '# of Macro Occurrences';
              VAR macro cnt;
             RUN;
       %END; %* if &alpha eq Y;
   %END; %* &nobs GT 0;
```

```
   %ELSE %DO; %* nobs eq 0;
       %* create temporary dataset to process in case there are no macros found;
       DATA work._temp2;
         macro2 = "No macros found";
       run;

       TITLE " Trace for &file";
       PROC PRINT DATA = work._temp2;
         VAR macro2;
       RUN;
       TITLE;
   %END; %* nobs eq 0;

   PROC DATASETS LIB = work
                 NOLIST;
     DELETE_temp:;
   QUIT;

   %GOTO finish;

   %nofile:
     %PUT &logfile does not exist;

   %finish:

%MEND trace;
```