Paper 097-2011

Conditional Processing Without Macros

John Ladds, Statistics Canada, Ottawa, Ontario, Canada

ABSTRACT

SAS® has some powerful tools to handle conditional program flow but you may not want all that power or the learning curve that goes with it. You may only want to turn on special code for debugging, error handling or in special cases. The RUN statement, with its optional argument CANCEL, can help create conditional processing routines without the need for more involved SAS tools.

INTRODUCTION

Why would anyone write a statement like RUN CANCEL? The CANCEL argument ends the current step without executing it. It really does not make any sense to use this statement but if you combine RUN and a macro variable you get the ability to control which steps run and which do not. Using the %LET statement and CALL SYMPUT function is the key.

DETAILS

Let's start with a simple example (Ref 01):

```
proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run cancel;
```

The log returns:

```
WARNING: The procedure was not executed at the user's request.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.04 seconds
      cpu time            0.03 seconds
```

Now let's add a simple way to control the program with a hard coded %LET statement. The following code will create a macro variable with the value CANCEL (Ref 02).

```
%let skipit = cancel;
proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run &skipit.;
```

When the program runs, the RUN &SKIPIT; statement will resolve to RUN CANCEL; and the PROC PRINT will not execute.

```
WARNING: The procedure was not executed at the user's request.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Using the OPTION SYMBOLGEN, we can see what is happening to the macro variable &skipit.

```
SYMBOLGEN:  Macro variable SKIPIT resolves to cancel
```

In the sample code, OPTION SYMBOLGEN is used to show what is happening with the macro variables. Normally this OPTION would be left at the default value NOSYMBOLGEN.

Changing the %LET code to

```
%LET SKIPIT=;
```

The RUN &SKIPIT; will resolve to RUN; and the PROC PRINT will execute;

```
NOTE: There were 15 observations read from the data set WORK.THEDATA.
NOTE: PROCEDURE PRINT used (Total process time):
      real time          0.00 seconds
      cpu time           0.01 seconds
```

Hard coding is all well and good but it is not really conditional processing. Another method to create macro variables is the CALL SYMPUT function. This function allows you to conditionally create or change a macro variable from within a DATA step. (Ref 03)

```
data _null_;
   set work.theData;
   if WTS eq 0
      then call symput('error','');
      else call symput('error',' cancel');
run;

proc print data=work.thedata;
   var ID Q01-Q03 WTS;
      run &error.;

proc print data=work.thedata;
   where WTS eq 0;
   var ID Q01-Q03 WTS;
      run &error.;
```

The DATA step can be used to decide whether or not to run the PROC PRINT depending on some condition. A WHERE statement on the PROC PRINT would only show the observations that meet the condition; not all the records.

It is also possible to have multiple macro variables defined on the RUN statement while debugging code, allowing some task to run but not during production or in the event of an error. (Ref 04)

```
%let debug=cancel;

proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run &error. &debug.;
```

But if both macro variables resolve to cancel at the same time the two cancels produce an error message in the program log. (Ref 5)

```
run cancel cancel;
233  run &error. &debug.;
```

```
SYMBOLGEN:  Macro variable ERROR resolves to cancel
SYMBOLGEN:  Macro variable DEBUG resolves to cancel
ERROR: Unrecognized form of the RUN statement.  Use either RUN; or RUN CANCEL;.
```

Other considerations when using RUN CANCEL are that it does not work with interactive procedures (i.e. "QUIT" rather than "RUN"), nor when in stream data is present in a step. Using CARDS, CARDS4, LINES, LINES4, DATALINES or DATALINES4 statements imply an end of step; i.e. "RUN" statement. The RUN CANCEL also writes a warning in the log which may not be acceptable in some situations.

CONCLUSION

RUN <CANCEL>; and macro variables are a simple way to control the flow of a program without the expense of learning SAS Macro Language or custom functions. The examples used in the paper are from the attached sample program and should work in most versions of SAS.

REFERENCES

SAS OnlineDoc 9.2. COPYRIGHT © 2002-2010, SAS INSTITUTE INC. CARY, NC, USA

CONTACT INFORMATION

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

Name:	John Ladds
Enterprise:	Statistics Canada – SED SAS Technology Centre
Address:	14-R-90 R.H. Coats Building, 100 Tunney's Pasture Driveway Ottawa, Ontario, Canada K1A 0T6
Phone:	613 951-1767
E-mail:	John.Ladds@statcan.gc.ca
Web:	www.statcan.ca

EXAMPLE CODE

```
dm log 'clear' log;
dm out 'clear' log;
ods noptitle;
options nomprint nomlogic nosymbolgen;
title1 'Conditional Processing without Macro Code';
title2 'Sample Code';

%let debug=;

data theData;
    input @01 id   $02.
          @03 Q01    1.
          @04 Q02    1.
          @05 Q03    1.
          @06 WTS    4.2;

 label ID  = 'Unique Identifier'
       Q01 = 'Test Question One'
       Q02 = 'Test Question Two'
       Q03 = 'Test Question Three'
       WTS = 'Weights';
 datalines4;
011230.23;
024560.34;
031230.45;
044560.10;
051230.29;
064560.38;
071230.47;
084560.56;
091230.65;
104560.74;
111230.83;
124560.92;
131230.01;
144560.12;
150000.00;
;;;;
run;

** Ref 1 Simple Example;

proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run cancel;

options symbolgen;

** Ref 2 Hard coded macro variables;

%let skipit = cancel;

proc print data=work.thedata;
```

Conditional Processing Without Macros, continued

```
   var ID Q01-Q03 WTS;
run &skipit.;

%let skipit =;

proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run &skipit.;

** Ref 3 Conditional macro variables;

%put Ref 3 Conditional macro variables;

data _null_;
   set work.theData;
   if WTS eq 0
      then call symput('error','');
      else call symput('error','cancel');
run;

proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run &error.;

** Ref 4 Multiple Conditional macro variables;

%let debug=;

data _null_;
   set work.theData;
   if WTS eq 0
      then call symput('error','');
      else call symput('error','cancel');
run;

proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run &error. &debug.;

** Ref 5 Multiple Conditional macro variables with errors;

%let debug=cancel;

data _null_;
   set work.theData;
   if WTS eq 0
      then call symput('error','');
      else call symput('error','cancel');
run;

proc print data=work.thedata;
   var ID Q01-Q03 WTS;
run &error. &debug.;

** The following RUN statement is only used to close the    ;
** PROC PRINT because of there intentional error in the demo;
run;
```