PhUSE 2009

Paper P015

CALL SYMPUT-Global or Local

Littish Dominic, Roche Products Limited, Welwyn Garden City, UK

ABSTRACT

CALL SYMPUT may not create a GLOBAL macro variable when defined inside a macro definition. When we use CALL SYMPUT inside a macro definition, it will write the macro variable values into a local symbol table, if the macro has a non empty local symbol table. If macro has an empty local symbol table, the macro variable defined inside a macro, using CALL SYMPUT routine has global scope. The rules do not apply when you explicitly define them as GLOBAL or LOCAL. This paper will discuss the scope of macro variable created using CALL SYMPUT and the reference of macro variable created by CALL SYMPUT with in the same data step.

INTRODUCTION

Macro variables comes in two varieties as local or global. A macro variable scope is local if it is defined inside a macro and Global if it is defined outside the macro definition.

CALL SYMPUT

CALL SYMPUT is a SAS language routine that assigns a value in a DATA step to a macro variable during execution time. It produces a global macro variable when its called inside a data step. This variable cannot be referenced inside the creating data step.

```
DATA _NULL_;
  CALL SYMPUT('_date', PUT(TODAY(),WORDDATE.));
RUN;
```

CALL SYMPUT produces global macro variable when used inside macro and the local symbol table is empty. For example when you invoke a SAS Session, the local symbol table will be empty.

```
%MACRO symp1;
  DATA _NULL_ ;
   CALL SYMPUT ('value1', '100');
  RUN;
%MEND symp1;

%symp1;

%PUT value is &value1;
```

The %PUT statement writes the following text to the SAS log

Value is 100

Now the VALUE1 has global scope because local symbol table is empty.

```
%MACRO symp2;
  %LOCAL Value2;
  DATA _NULL_;
    CALL SYMPUT ('value2', '100');
    CALL SYMPUT ('value3', '200');
  RUN;
%MEND;

%symp2;

%PUT value2 is &value2;
%PUT value3 is &value3;
```

The %PUT statement writes the following text to the SAS log in this scenario,

```
WARNING: Apparent symbolic reference VALUE2 not resolved.
WARNING: Apparent symbolic reference VALUE3 not resolved.
```

In this case, VALUE2 is defined as local. But both VALUE2 and VALUE3 became local variable because when 'VALUE2' creates the local symbol became non empty. If the macro contains positional or keyword parameter(s), the macro variable created by CALL SYMPUT has a local scope. This is because the local symbol table is non empty (since the local symbol table contains the keyword/positional parameter.)

If the macro contains macro variables created using SYSPBUFF or contains %GOTO statement, CALL SYMPUT creates macro variable in the local symbol table. If CALL SYMPUT comes after a PROC SQL with INTO clause in a macro, it has local scope.

```
%MACRO sysp1/PARMBUFF;
  %PUT SYSPBUFF (parameters) contains: &SYSPBUFF;
   DATA symp3;
     CALL SYMPUT("value4", 400);
   RUN;
%MEND;

%sysp1(age, height, weight);

%PUT value4 is &value4;
```

The %PUT statement writes the following text to the SAS log in this scenario,

```
WARNING: Apparent symbolic reference VALUE4 not resolved.
```

]REFERENCE THE MACRO VARIABLE CREATED USING CALL SYMPUT WITH IN THE SAME DATA STEP

The macro variable created by CALL SYMPUT can not be referenced inside the creating data step. But CALL EXECUTE, SYMGET and RESOLVE can be used to reference a macro variable with in the data step.

CALL EXECUTE:

CALL EXECUTE statements get resolved first and then moved to the input stack while iterating the data step. If the argument is a SAS statement then it gets moved to the input stack immediately and starts to execute at the data step boundary

```
DATA _NULL_ ;
   CALL SYMPUT ('nofobs', 100);
   CALL EXECUTE('%put The no of observation is & nofobs; ') ;
RUN;
```

The %PUT statement writes the following text to the SAS log

The number of observation is 100

SYMGET:

SYMGET returns the value of macro variable during execution time.

```
%MACRO symg1;
    DATA symp3;
      CALL SYMPUT("value4", 400);
      value4=SYMGET("value4")*2;
      PUT "Value4 is " value4;
    RUN;
%MEND;

%symg1;
```

The %PUT statement writes the following text to the SAS log

Value4 is 800

RESOLVE:

This function resolves the value of a text expression during DATA step execution

```
%MACRO resolve1;
    DATA symp3;
      CALL SYMPUT("value5", 400);
      Value5=RESOLVE(&value5)*2;
      PUT "Value5 is " value5;
    RUN;
%MEND;

%resolve1;
```

The %PUT statement writes the following text to the SAS log

Value5 is 400.

CONCLUSION

This paper discussed scope of macro variable created using CALL SYMPUT and how it can be resolved with in a same data step. It is hoped that reading this paper enables users to better understand SAS system processing and thus employ the SAS system more effectively in the future.

CONTACT INFORMATION

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

Author Name: Littish Dominic
Company: Roche Products Limited
Address: 6, Falconway
City / Postcode: AL7 1TW
Work Phone: +44 170736 5692
Email: littish.dominic@roche.com
Web: www.roche.com