PhUSE 2015

Paper CC08

The Ampersand (&) Challenge, Single, Double or more?

Amar Nayak, Freelance SAS® Consultant, Maidenhead, UK

ABSTRACT

The resolution of a macro variable requires a pre-fixed "&" (ampersand). Most SAS® Programmers encounter the use of macro variable(s) in the basic form e.g "&macrovar" by defining the macro variable in various ways (precisely five). However, as the degree of complexity of any SAS® macro program increases, the use of multiple ampersands plays an important role to build the flow of the macro program, more efficient and robust. Such opportunities don't occur frequently unless your programming role has the potential to deal with frequent SAS® development projects. Less experienced SAS® programmers find it a challenge to understand the resolution of multiple ampersands in a simplistic way. The path to glory is thorough understanding of multiple ampersands resolution process when it is used in SAS® macro program. This paper attempts to provide a detailed insight into the macro variable resolution process and explain it in layman's term with logical examples.

PURPOSE OF THIS PAPER

As a programmer gains experience building macro code, it becomes easier to deal with the basic macro variable concepts in terms of its creation and use. However, how many of us are then ready to take the plunge of macro glory i.e by taking the "Multiple Ampersand Challenge"? An analogy to such an ampersand challenge is the anxiety some parents have when thinking of a second child once they have already settled with one, SCARY! Therefore, just as any family counsellor comforts you that once dealt with the first one successfully, the second, third, fourth and so on become much easier to handle because of maturity and experience, the intention of this paper is to relay the same message for handling multiple ampersand challenge and not children of course! ☺

Furthermore, in pure technical terms, macro variables with greater than one ampersand require multiple scans by the macro processor. The processor scans and resolves references for the length of the variable (usually delimited by a special character or semicolon) until all references have been resolved. The details and understanding of this macro resolution process is not understood easily. Therefore, the further aim of this paper is to interpret this technical definition through a graphical representation of a macro resolution process that any SAS® Programmer shall easily understand.

In order to aid the understanding of the "Multiple Ampersand Challenge", this paper starts with a simple scenario moving on to a more complex scenario with a number of examples.

RULES OF THE GAME

To deal with any number of ampersands associated with a macro variable, the three basic rules that should be understood and these are:

a) The macro processor reads a macro variable from left to right (i.e. forward scanning).
b) For multiple ampersands, the re-scan rule takes effect until no more ampersands are left to resolve i.e the macro variable resolves its final value OR error messages have been issued
a) 'Two' ampersands result into 'One' ampersand (&& ➔ &) during the macro resolution process

HANDLING MULTIPLE AMPERSANDS

SCENARIO 1

To begin with, let me illustrate an example of defining four macro variables separately and how multiple ampersands deal with these uniquely defined macro variables [denoted by **(n)**].

```
options symbolgen;

%LET BOOK  = STORY;    **(1)**;
%LET STORY = FUNNY;    **(2)**;
%LET FUNNY = TWIST;    **(3)**;
%LET TWIST = CLIMAX;   **(4)**;
```

```
%put &BOOK;
%put &&&BOOK;
%put &&&&&&&BOOK;
%put &&&&&&&&&&&&&&&BOOK;
```

Y	M	□	R
1	&BOOK	→	STORY
2	&&BOOK	→	STORY
3	&&&BOOK	→	FUNNY
4	&&&&BOOK	→	STORY
5	&&&&&BOOK	→	FUNNY
6	&&&&&&BOOK	→	FUNNY
7	&&&&&&&BOOK	→	TWIST
8	&&&&&&&&BOOK	→	STORY
9	&&&&&&&&&BOOK	→	FUNNY
10	&&&&&&&&&&BOOK	→	FUNNY
11	&&&&&&&&&&&BOOK	→	TWIST
12	&&&&&&&&&&&&BOOK	→	FUNNY
13	&&&&&&&&&&&&&BOOK	→	TWIST
14	&&&&&&&&&&&&&&BOOK	→	TWIST
15	&&&&&&&&&&&&&&&BOOK	→	CLIMAX

Y: Number of ampersands pre-fixed to the macro variable
M: Macro variable to be resolved
R: Final resolved value

Look closely at the marked rows above. Based on the number of ampersands (**Y**) and macro variables [denoted by ****(n)****, see page 1], a non-linear equation can be drawn between them as follows (The equation is specific to this example discussed).

$$y = 2^{n} - 1$$

In mathematical terms,

- 'n' is the number of macro variables that are defined
- 'y' is the number of ampersands that are required until the macro variable successfully resolves to its final value

The number of iterations required until the macro variable successfully resolves to its final value is the same as the number of macro variables defined. Now, to begin with

(1) &BOOK → STORY

After executing the macro variable, the SAS® log will appear as follows

```
31          %put &BOOK;
SYMBOLGEN:  Macro variable BOOK resolves to STORY
STORY
```

➡ One iteration to its final value highlighted

This example is a straightforward 'no brainer' macro resolution.

(2) **&&&BOOK** ① ➔ **&STORY** ② ➔ **FUNNY**

In this example, based on the macro variable resolution rules mentioned earlier (see page 1), the SAS® log appears as follows

```
32          %put &&&BOOK;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable BOOK resolves to STORY
```

➔ After 1st iteration

① During the first read of the macro variable from left to right, first two ampersands result into one (**&&** ➔ **&**) and the rest **&BOOK** resolves to its defined value as **STORY**

```
SYMBOLGEN:  Macro variable STORY resolves to FUNNY
FUNNY
```

➔ After 2nd iteration to its final value highlighted

② During the second read, **&STORY** resolves to its defined value as **FUNNY**

(3) **&&&&&&&BOOK** ① ➔ **&&&STORY** ② ➔ **&FUNNY** ③ ➔ **TWIST**

In this example, the SAS® log appears as follows

```
33          %put &&&&&&&BOOK;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable BOOK resolves to STORY
```

➔ After 1st iteration

① During the first read of the macro variable from left to right, three pairs of double ampersands result into one (**&&** ➔ **&**) and the rest **&BOOK** resolves to its defined value as **STORY**

```
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable STORY resolves to FUNNY
```

➔ After 2nd iteration

② During the second read, first two ampersands result into one (**&&** ➔ **&**) and the rest **&STORY** resolves to its defined value as **FUNNY**

```
SYMBOLGEN:  Macro variable FUNNY resolves to TWIST
TWIST
```

➔ After 3rd iteration to its final value highlighted

③ During the third read, **&FUNNY** resolves to its defined value as **TWIST**

(4) **&&&&&&&&&&&&&&&BOOK** ① ➔ **&&&&&&STORY** ② ➔ **&&&FUNNY** ③ ➔ **&TWIST** ④ ➔ **CLIMAX**

In this example after executing the macro variable, the SAS® log will appear as follows

```
34          %put &&&&&&&&&&&&&&&BOOK;
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
```

```
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable BOOK resolves to STORY
```

➡ After 1st iteration

① During the first read of the macro variable from left to right, seven pairs of double ampersands result into one (&& ➔ &) and the rest &BOOK resolves to its defined value as STORY

```
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable STORY resolves to FUNNY
```

➡ After 2nd iteration

② During the second read, three pairs of double ampersands result into one (&& ➔ &) and the rest &STORY resolves to its defined value as FUNNY

```
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable FUNNY resolves to TWIST
```

➡ After 3rd iteration

③ During the third read, first two ampersands result into one (&& ➔ &) and the rest &FUNNY resolves to its defined value as TWIST

```
SYMBOLGEN:  Macro variable TWIST resolves to CLIMAX
CLIMAX
```

➡ After 4th iteration to its final value highlighted

④ During the fourth read, &TWIST resolves to its defined value as CLIMAX

SCENARIO 2

The second scenario illustrates how multiple ampersands help to build efficient and robust macro code.

Example 1:

The goal is to list city (country) and year where past PhUSE conferences have been held since 2008. To begin with, we define macro variables separately for city, country and year value to output past conference details. This method is termed as 'direct macro reference' that requires just one ampersand for macro variable resolution.

```
*****************************;
*** DEFINE MACRO VARIABLES ***;
*****************************;
%let phuscity1 = Manchester;
%let phuscntr1 = England;
%let yr1 = 2008;

%put PhUSE Conference was held at &phuscity1 (&phuscntr1) in &yr1;

%let phuscity2 = Basel;
%let phuscntr2 = Switzerland;
%let yr2 = 2009;

%put PhUSE Conference was held at &phuscity2 (&phuscntr2) in &yr2;
```

Here the conference details of any year gets displayed through a combination of free text and directly referenced macro variables

A snippet of the SAS® log after executing the SAS® program above is as follows

```
SYMBOLGEN:  Macro variable PHUSCITY1 resolves to Manchester
SYMBOLGEN:  Macro variable PHUSCNTR1 resolves to England
SYMBOLGEN:  Macro variable YR1 resolves to 2008
PhUSE Conference was held at Manchester (England) in 2008
```

```
SYMBOLGEN:  Macro variable PHUSCITY2 resolves to Basel
SYMBOLGEN:  Macro variable PHUSCNTR2 resolves to Switzerland
SYMBOLGEN:  Macro variable YR2 resolves to 2009
PhUSE Conference was held at Basel (Switzerland) in 2009
```

Example 2:

With the same goal to list city (country) and year where past PhUSE conferences have been held since 2008, 'indirect macro variable' reference is used by creating macro variables that belong to a series of macro variables to output the past conference details via a %DO loop. The macro facility provides indirect macro variable referencing which allows us to use an expression (e.g YR&I) to generate a reference to one of a series of macro variables. For e.g, the value of macro variable 'I' could be used to reference a variable in the series of macro variables named YR1 to YR7. The following example provides greater insight in understanding this programming approach.

```
*************************;
*** CREATING DUMMY DATA ***;
*************************;
data phuse_city;
  input yr city $11. country $12. @;

  datalines;
       2008 Manchester England
       2009 Basel      Switzerland
       2010 Berlin     Germany
       2011 Brighton   England
       2012 Budapest   Hungary
       2013 Brussels   Belgium
       2014 London     England
       2015 Vienna     Austria
       ;
run;

*****************************;
*** DEFINE MACRO VARIABLES ***;
*****************************;
data _null_;
  set phuse_city end=eod;

  /* CREATE MACRO VARIABLES STORING START YEAR VALUE */
  if _n_ = 1 then call symput("startyr", compress(put(yr, best.)));

  /* CREATE MACRO VARIABLES STORING CURRENT YEAR VALUE */
  if eod then call symput("curryr", compress(put(yr, best.)));

  /* CREATE FIRST SET OF EIGHT MACRO VARIABLES STORING YEAR VALUES */
  call symput("yr"||compress(put(yr, best.)), compress(put(yr, best.)));

  /* CREATE SECOND SET OF EIGHT MACRO VARIABLES STORING CITY VALUES
     CORRESPONDING TO EACH YEAR */
  call symput("phuscity"||compress(put(yr, best.)), compress(city));

  /* CREATE THIRD SET OF EIGHT MACRO VARIABLES STORING COUNTRY
     VALUES CORRESPONDING TO EACH CITY */
  call symput("phuscntr"||compress(put(yr, best.)), compress(country));

run;
```

The macro variables (and series) defined are
1) &startyr
2) &curryr
3) &yr2008 - &yr2015
4) &phuscity2008 - &phuscity2015
5) &phuscntr2008 - &phuscntr2015

```
*************************************************************************;
*** CREATE A MACRO CODE WITH %DO LOOP TO OUTPUT PAST CONFERENCE DETAILS ***;
*** USING MACRO VARIABLES ASSOCIATED WITH DOUBLE AMPERSANDS (&&)        ***;
*************************************************************************;
%macro phuse_city ();

%do i = &startyr %to &curryr;

  %put The PhUSE Conference was held at &&phuscity&i (&&phuscntr&i) in
       &&yr&i ;

%end;

%mend phuse_city;
```

Here the conference details of any year gets displayed through a combination of free text and indirectly referenced macro variables

```
**********************;
*** EXECUTE MACRO ***;
**********************;
%phuse_city;
```

A snippet of the SAS® log after executing the improvised SAS® program is as follows

```
MLOGIC(PHUSE_CITY):  Beginning execution.
SYMBOLGEN:  Macro variable STARTYR resolves to 2008
SYMBOLGEN:  Macro variable CURRYR resolves to 2015
MLOGIC(PHUSE CITY):  %DO loop beginning; index variable I; start value is 2008; stop value is 2015;
MLOGIC(PHUSE CITY):  %PUT The PhUSE Conference was held at &&phuscity&i (&&phuscntr&i) in &&yr&i
SYMBOLGEN:  && resolves to &.                                                                  (1)
SYMBOLGEN:  Macro variable I resolves to 2008
SYMBOLGEN:  Macro variable PHUSCITY2008 resolves to Manchester                                 (2)
SYMBOLGEN:  && resolves to &.                                                                  (1)
SYMBOLGEN:  Macro variable I resolves to 2008
SYMBOLGEN:  Macro variable PHUSCNTR2008 resolves to England                                    (2)
SYMBOLGEN:  && resolves to &.                                                                  (1)
SYMBOLGEN:  Macro variable I resolves to 2008
SYMBOLGEN:  Macro variable YR2008 resolves to 2008                                             (2)
The PhUSE Conference was held at Manchester (England) in 2008
```

As you can observe from the SAS® log, the macro variable associated with the value of city, country and year takes two iterations to output its final value. Macro variable resolution rules mentioned earlier (on page 1) come into effect before the entire conference text output gets displayed at the end.

Example 3:

There is potential to further improvise the SAS® macro program above by using multi macro variable combination along with the use of %DO loop to output past conference details. This example deals with multiple ampersands (> 2).

```
*************************************************************;
*** ADDITIONAL VARIABLE IN DUMMY DATA (CONTINUED FROM ABOVE) ***;
*************************************************************;
data phuse_city;
  set phuse_city;

  conftext = "PhUSE Conference was held at "||compress(city)||"
             ("||compress(country)||") in "||compress(put(yr, best.));

run;
```

conftext
PhUSE Conference was held at Manchester (England) in 2008
PhUSE Conference was held at Basel (Switzerland) in 2009
PhUSE Conference was held at Berlin (Germany) in 2010
PhUSE Conference was held at Brighton (England) in 2011
PhUSE Conference was held at Budapest (Hungary) in 2012
PhUSE Conference was held at Brussels (Belgium) in 2013
PhUSE Conference was held at London (England) in 2014
PhUSE Conference was held at Vienna (Austria) in 2015

```
*************************************************************;
*** DEFINE ADDITIONAL MACRO VARIABLE (CONTINUED FROM ABOVE) ***;
*************************************************************;
data _null_;
  set phuse_city;

  /* CREATE ADDITIONAL SET OF EIGHT MACRO VARIABLES STORING PAST CONFERENCE
     DETAILS */
  call symput(compress(city)||compress(country)||compress(put(yr, best.)),
              strip(conftext));

run;
```

```
The macro variables defined are
1)  &MANCHESTERENGLAND2008
2)  &BASELSWITZERLAND2009
3)  &BERLINGERMANY2010
4)  &BRIGHTONENGLAND2011
5)  &BUDAPESTHUNGARY2012
6)  &BRUSSELSBELGIUM2013
7)  &LONDONENGLAND2014
8)  &VIENNAAUSTRIA2015
```

```
*************************************************************************;
*** CREATE A MACRO CODE WITH %DO LOOP TO OUTPUT PAST CONFERENCE DETAILS ***;
*** THROUGH MULTI MACRO VARIABLE COMBINATION PREFIXED BY MULTIPLE &     ***;
*************************************************************************;
%macro phuse_city_loop ();

%do i = &startyr %to &curryr;

  %put &&&&&&&&&&&&&&&&&&phuscity&i..&&phuscntr&i..&&yr&i..;

%end;

%mend phuse_city_loop;
```

Here the conference details of any year gets displayed with mutli macro variable combination (through indirect macro reference) without the use of any free text

```
**********************;
*** EXECUTE MACRO ***;
**********************;
%phuse_city_loop;
```

A snippet of the SAS® log after executing the SAS® macro appears as follows

```
MLOGIC(PHUSE_CITY_LOOP):  Beginning execution.
SYMBOLGEN:  Macro variable STARTYR resolves to 2008
SYMBOLGEN:  Macro variable CURRYR resolves to 2015
MLOGIC(PHUSE_CITY_LOOP):  %DO loop beginning; index variable I; start value is 2008; stop value is 2015;
MLOGIC(PHUSE_CITY_LOOP):  %PUT &&&&&&&&&&&&&&&&&&phuscity&i..&&phuscntr&i..&&yr&i..
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable I resolves to 2008
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable PHUSCITY2008 resolves to Manchester
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable I resolves to 2008
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable PHUSCNTR2008 resolves to England
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable I resolves to 2008
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable YR2008 resolves to 2008
SYMBOLGEN:  Macro variable MANCHESTERENGLAND2008 resolves to
PhUSE Conference was held at Manchester (England) in 2008
```

First Iteration

Second Iteration

Third Iteration

Fourth Iteration

Final Value

As you can observe from the SAS® log, after five iterations and orderly resolution of macro variables, the final conference text output is successfully displayed. However this is just one of the eight values programmed for display. The %DO loop will continue to execute until the current year value (2015) passes through the loop, to output conference text value for 2015.

CONCLUSION

Handling of macro variables with leading multiple ampersands in any SAS® macro program requires careful planning to get the desired results. If structured wisely, it can considerably reduce the code length resulting in an efficient and robust program that can handle changes to the final output requirements with minimal updates to the existing macro program. I do hope this encourages all SAS® programmers to crack this ultimate puzzle and take the 'Multiple Ampersand Challenge'.

REFERENCES

http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#a001071915.htm

http://www2.sas.com/proceedings/sugi29/063-29.pdf

http://www.okstate.edu/sas/v8/saspdf/macro/c03.pdf

ACKNOWLEDGMENTS

I would like to sincerely thank the individuals below for their thorough review of this paper that has helped me in my diligent attempt to explain an intricate topic in SAS® macro programming

- Stephen Gormley (Biostatistical Programming Senior Manager, Amgen)
- Vishwas Jadhav (Manager Statistical Programming, INC Research)
- Shambhavi Nayak (Clinical Programmer, PRA-HS)

CONTACT INFORMATION

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

Amar Nayak
Freelance SAS® Consultant
13, Bridge Court, Bridge Avenue,
Maidenhead, SL6 1RW,
Berkshire, UK
Web: www.sequensysltd.com
E-mail: amarsnayak@gmail.com
LinkedIn: https://uk.linkedin.com/in/amarsnayak