Macro Quoting Functions, Other Special Character Masking Tools, and How To Use Them

Arthur L. Carpenter
California Occidental Consultants

ABSTRACT
Quoting functions allow the user to pass macro arguments while selectively removing the special meaning from characters such as &, %, ;, ‘, and “. Most of these functions are **not** commonly used and are even less commonly understood. Although they are powerful and can even be necessary, usually programming solutions are available that do not require the use of the quoting functions.

When are quoting functions needed? When needed how are they used? Which one should be selected? This paper will discuss when to use and when to avoid quoting functions. In addition the discussion will include solutions that avoid the use of quoting functions. Fortunately there are several ways to mask special characters within the macro language. These include quoting functions, %DO blocks, restructuring of statements, and the character mask (%).

KEY WORDS
macro, quoting functions, masking, special characters

INTRODUCTION
Certain characters or combinations of characters will often cause the macro language to behave in ways that are neither desirable or anticipated. The programmer writing the SAS® macro must anticipate these problems or code so that problems will not occur. Macro coding techniques, character masks, and macro quoting functions all play a part in creating robust macro code. Many of the examples and some of the text in this paper are based on sections in *Carpenter’s Complete Guide to the SAS® Macro Language*.

CODING TO AVOID QUOTING
The most commonly used macro quoting function is %STR. Often it is used along with the %LET statement to mask semicolons that would otherwise terminate the %LET.

In the following example we want to create a macro variable &P that contains two SAS statements;

```
%LET P=PROC PRINT DATA=DSN; RUN;;
```

Because the semicolon following DSN terminates the %LET statement, the macro variable &P contains:

```
PROC PRINT DATA=DSN
```

which results in a syntax error for the missing semicolon.

The %STR function masks the semicolon by quoting it.

```
%LET P=%STR(PROC PRINT DATA=DSN;
            RUN;);
```

This results in the macro variable &P being correctly assigned the two statements.

```
PROC PRINT DATA=DSN; RUN;
```

In the following macro the %STR function is used because multiple statements (with semicolons) are needed in the DATA _NULL_ step.

```
%macro exist(dsn);
%global exist;
%if &dsn ne %then %str(
   data _null_;
   if 0 then set &dsn;
   stop;
   run;
   );
```

```
%if &syserr=0 %then %let
exist=yes;
%else %do;
   %let exist=no;
   %put PREVIOUS ERROR USED TO
CHECK FOR PRESENCE ;
   %put OF DATASET & IS NOT A
PROBLEM;
%end;
%mend exist;
```

In the above macro a %DO could have been used instead of the %STR function. This results in code that I believe is easier to understand.

```
%if &dsn ne %then %do;
   data _null_;
   * No observations are actually
read;
   if 0 then set &dsn;
   stop;
   run;
%end;
```

As a general rule I would rather explore alternate coding methodologies like this one before resorting to quoting functions. Another alternate to quoting functions in some circumstances is the use of the % masking character.

MASKING SPECIAL CHARACTERS

A number of symbols are usually expected in pairs. These include the ‘, “, (, and). Errors will usually be created if only one half of the pair is specified in a string. This can especially be a problem if you want to include one of these mismatched symbols in a macro variable.

Some of the quoting functions expect these symbols to be in pairs *e.g.* %STR, others allow them to be unpaired *e.g.* %BQUOTE. When you need to use a mismatched symbol where it will otherwise cause a problem, you can precede the mismatched symbol with a % to mask its meaning.

In the following example we want to assign the value (abcd to the macro variable &B. The LOG shows that the %STR is incorrectly constructed in line 40. The second open parenthesis causes the %STR to be closed by the) on line 41.

```
40   %let b = %str((abcd);
41   * Unclosed );
42
43   %put &b ;
(abcd); * Unclosed
```

This is corrected by using the % to mask the meaning of the second (.

```
44   %let b = %str(%(abcd);
45   * Unclosed );
46
47   %put &b ;
(abcd
```

A table showing several examples of the use of the % to mask characters can be found in Table 7.2 on p. 81 in *SAS® Macro Language: Reference, First Edition* and in Carpenter(1998, p. 94).

QUOTING FUNCTIONS

Quoting functions operate by adding what is essentially invisible characters before and after the string which is to be quoted. Once added these characters remain until stripped off by the macro processor (%UNQUOTE or by passing the text to the SAS System for processing).

There are two issues that make quoting functions more difficult to understand. These have to do with functions that:

- either do or don’t rescan the text for % and & references
- are executed at statement compilation or at statement execution

Rescan and NoRescan

Of special interest in the text strings to be quoted are those that contain the special macro symbols % and &. Normally these symbols indicate references to macro calls and macro variables that must be resolved prior to the execution of the function. These references are resolved by rescanning the text as many times as it takes to resolve the usage of the % and &. Sometimes you do not want these references

resolved (you want to pass the macro call or macro variable without resolution). To do this you need a function designated as a NoRescan function (NR).

Many of the quoting functions come in pairs (the name either does or does not start with the letters NR *e.g.* %STR and %NRSTR. The two functions will perform essentially the same operation except the one with the NR will also remove the meaning from the % and &. When you hear the jargon ' the meaning is removed', this means that the macro processor will not 'see' the % and & as the special characters that it normally does.

Compilation / Execution

Since macro statements are compiled and then executed you may need to control when the text string is to be quoted. Some quoting functions remove the meaning during compilation while others remove it during macro execution. When meaning is removed during compilation (%STR and %NRSTR) the resultant text is passed to the processor, but not the function call. The other quoting functions are resolved during the execution of the macro. For these functions the entire call to the function and its text is passed to the macro processor where it is resolved.

This will rarely be an issue for most programmers. Usually you only will need to worry if text that is resolved during execution becomes syntactically incorrect or misinterpreted. Consider the following short example:

```
%let type = or;
%if &type ne xx %then %do;
```

The macro variable &TYPE is resolved during the evaluation of the expression which results in:

```
%if or ne xx %then %do;
```

The OR is seen as the logical mnemonic operator and an error message is produced. If &TYPE is quoted during execution the resolved OR will be treated as text and there will not be a problem with the syntax.

```
%let type = or;
%if %quote(&type) ne xx %then %do;
```

QUOTING FUNCTION OVERVIEW

The following table gives a fairly brief overview of the quoting functions and their behavior.

The quoting functions are:

%BQUOTE

removes meaning from unanticipated special characters (except & and %) during execution.

%NRBQUOTE

removes meaning from unanticipated special characters including & and % during execution.

%QUOTE

removes meaning from a string (except % and &) during execution.

%NRQUOTE

removes meaning from %, &, special characters, and mnemonics during execution.

%STR

removes meaning from special characters (except % and &) at compilation.

%NRSTR

removes meaning from special characters including % and & at compilation.

%SUPERQ

prevents any resolution of the value of a macro variable

%UNQUOTE

undoes quoting.

USING NORESCAN

The %NRSTR function behaves in the same way as %STR except meaning is also removed from the % and &. Macro variable references in the %STR are resolved.

```
%LET CITY = MIAMI;
%PUT %STR(&CITY) IS ON THE WATER.;
```

The LOG would show:

```
MIAMI IS ON THE WATER.
```

When %NRSTR is used instead of %STR, the macro variable &CITY is not resolved because the special meaning has been removed from the &.

```
%LET CITY = MIAMI;
%PUT %NRSTR(&CITY) IS ON THE
WATER.;
```

The LOG would show:

```
&CITY IS ON THE WATER.
```

Depending on how the string containing the unresolved &CITY is used it might cause errors or warnings.

BLIND QUOTES

Blind quoting functions are used to remove meaning from unanticipated characters during macro execution. It is especially useful if the text string was entered by a user through an application and you may not have been able to trap all possible characters that might cause the macro to fail.

Assume that the macro variable &METHOD has been assigned the value:

THE DOCTOR'S NEW THERAPY.

It is very likely that when &METHOD is resolved the SAS System would produce an error because of the unmatched single quote ('). This is demonstrated in the following %LET. When it is executed the mismatched quote will mask the semicolon and will almost certainly create syntax problems.

```
%let method2 = &method;
```

When resolved this statement becomes:

```
%let method2 = THE DOCTOR'S NEW
THERAPY;
```

To get around this problem, %BQUOTE could be used as follows:

```
%let method2=%BQUOTE(&method);
```

The special meaning will be removed from the single quote when resolved and it will therefore not cause syntax problems.

The %NRBQUOTE function is similar to %BQUOTE except the meaning is also removed from % and & after resolution.

UNQUOTING

Once a quoting function has been applied its effects remain associated with the text (even in subsequent usages). If you need to remove or change the effects of any of the other quoting functions, the %UNQUOTE is used.

Three macro variables are defined below, but the second, &OTH, is defined using the %NRSTR function. This means that &CITY can not be resolved when &OTH is resolved. When the %UNQUOTE function is applied to &OTH its value (&CITY) is seen as a macro variable which is also resolved.

```
%let city = miami;
%let oth = %nrstr(&city);
%let unq = %unquote(&oth);

%put &city &oth &unq;
```

The LOG shows:

```
miami &city miami
```

Although &OTH looks like any other macro variable in the %PUT statement, it will not be treated as such because it is quoted. This can cause programming problems if the programmer does not know that a macro reference or special character has been quoted.

OTHER MACRO FUNCTIONS THAT QUOTE

Macro functions are very analogous to DATA step functions and perform similar text operations. In addition several macro functions have counterparts that return quoted results. Like the LEFT and SUBSTR data step functions, the %QLEFT(and %LEFT) and %QSUBSTR (and %SUBSTR) macro functions perform similar text manipulations.

The macro %CMPRES which removes multiple blanks as well as leading and trailing blanks is supplied with the SAS System and is shown *sans comments* below.

```
%macro cmpres(text);
%local i;
%let i=%index(&text,%str(  ));
%do %while(&i^=0);
  %let text=
    %qsubstr(&text,1,&i)%qleft(
       %qsubstr(&text,&i+1));
  %let i=%index(&text,%str(  ));
%end;
%left(%qtrim(&text))
%mend;
```

The macro is written to protect the user from text results that could cause problems. The macro uses %QLEFT, %QSUBSTR, and %QTRIM to quote the results of the various operations within the macro.

SUMMARY

It is not necessary to have a complete understanding of each of these functions but rather a general understanding of what they do. Some of these functions will be used much more often than others. It has been this author's experience that the %STR function is the most widely used quoting function.

ABOUT THE AUTHOR

Art Carpenter's publications list includes two chapters in *Reporting from the Field*, the two books *Quick Results with SAS/GRAPH® Software*, and *Carpenter's Complete Guide to the SAS® Macro Language*, and over two dozen papers and posters presented at SUGI, PharmaSUG, and WUSS. Art has been using SAS since 1976 and has served as a steering committee chairperson of both the Southern California SAS User's Group, SoCalSUG, and the San Diego SAS Users Group, SANDS; a conference cochair of the Western Users of SAS Software regional conference, WUSS; and Section Chair at the SAS User's Group International conference, SUGI.

Art is a SAS Quality Partner™ and through California Occidental Consultants he teaches SAS courses and provides contract SAS programming support nationwide.

Art Carpenter
California Occidental Consultants
P.O. Box 6199
Oceanside, CA 92058-6199

(760) 945-0613
art@caloxy.com
www.caloxy.com

REFERENCES

Carpenter, Arthur L., *Carpenter's Complete Guide to the SAS® Macro Language*, Cary, NC:SAS Institute Inc., 1998, 242 pp.

SAS® Macro Language: Reference, First Edition, Cary, NC:SAS Institute Inc., 1997, 304 pp.

TRADEMARK INFORMATION