Macro Quoting Functions

Jerry Mock
SAS Institute Inc.

Quoting Functions Available in the SAS Macro Language

Compile time functions

- %STR(argument)
- %NRSTR(argument)

Execution time functions

- %QUOTE(argument)
- %BQUOTE(argument)
- %NRQUOTE(argument)
- %NRBQUOTE(argument)
- %UNQUOTE(argument)

*new with Version 5 of the SAS System

Why do we need Quoting Functions?

Definition: Token

A token is a sequence of characters that is treated as a single data item.

There is a class of characters that are tokens and that the SAS macro language uses these characters for its lexical scan.

These characters are typically called the "special characters."

A partial list follows:

<table>
<thead>
<tr>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>semicolon</td>
</tr>
<tr>
<td>,</td>
<td>comma</td>
</tr>
<tr>
<td>+ -</td>
<td>arithmetic</td>
</tr>
<tr>
<td>/ *</td>
<td>operators</td>
</tr>
<tr>
<td>= &lt;</td>
<td>comparison</td>
</tr>
<tr>
<td>&gt;</td>
<td>operators</td>
</tr>
<tr>
<td>&amp;</td>
<td>logical</td>
</tr>
<tr>
<td>(</td>
<td>left parenthesis</td>
</tr>
<tr>
<td>)</td>
<td>right parenthesis</td>
</tr>
</tbody>
</table>

The following "special characters" must be paired

- single quote literal definition
- double quote literal definition

If any of the above tokens are to be treated as part of a string and not part of the syntax, a facility must be made available. The quoting functions of the SAS macro language constructs tokens that can contain any of the above special characters.

There are three components of the SAS macro language that use the above characters to direct their action. The components are the macro compiler, the macro executor, and the string evaluator (%EVAL).

Compile time functions

- %STR(argument)
- %NRSTR(argument)

The macro language is a text string processor.

Some characters that may appear in a text string have meaning to the macro compiler.

The following example illustrates a conflict of meaning:

```
%MACRO FREQ(T);
  %IF &T,= %THEN PROC FREQ; TABLE &T;
  %ELSE %PUT FREQ MACRO REQUIRES A VARIABLE;
%MEND FREQ;

%FREQ(AGE)
```

In the above example we wish to emit

"PROC FREQ; TABLE AGE;"

to the SAS System. However, the macro is compiled and stored as:

```
%IF &T,= %THEN PROC FREQ; TABLE &T;
%ELSE %PUT FREQ MACRO REQUIRES A VARIABLE;
```

When invoked, the SAS System will see

```
PROC FREQ
  TABLE AGE;
```

The semicolon is missing from the PROC FREQ statement. When coding the macro FREQ we need a way of masking the meaning of the semicolon from the macro compiler so it can be passed to the SAS System.

Masking the meaning of characters

There are two macro functions that mask the meaning of special characters at compile time. They are

- %STR(argument)
- %NRSTR(argument)

These functions allow you to use characters that have a special meaning to the macro language without invoking their special meaning.
%STR masks the meaning of
blanks leading and trailing blanks
) ( parentheses
/ * + - = ** mathematical operators
< > = - comparison operators
LE GE LT GT comparison operators
EQ NE | AND OR NOT logical operators
; , delimiters

%NRSTR masks the meaning of
blanks leading and trailing blanks
) ( parentheses
/ * + - = ** mathematical operators
< > = - comparison operators
LE GE LT GT comparison operators
EQ NE | OR & AND NOT logical operators
; , delimiters
& % macro triggers
&nnn %nnn symbolic resolution and macro invocation

Note that the functions perform their task at compile time and are not available at execution time.

Masking the meaning of special characters
When the SAS macro language encounters %STR or %NRSTR the following occurs:

before invoking after invoking
the function the function

%STR(PROC SORT;); | -------> |&; PROC SORTA;
) | perform |&
| | | |
%STR(&X+&Y) + | function |& &X+&Y
- | | &-
| | | |
%NRSTR(&X+&Y) \ | & &X+&Y
C | -------> |C
| | |
O | | O
P | | P
R | | R
S | | S
T | -------> |T

The delta function takes the special characters and removes their macro meaning.

The new characters generated are not identical to the old characters as far as the word scanner is concerned but %EVAL compares them equal.

Tokenizing within %STR
The SAS System word scanner sees all characters before the macro facility. As the word scanner processes characters, it establishes rules for the next character or group of characters.

What are the wordscanner's rules for the following examples?

Example 1.
%let y=left parenthesis ();
%put &y;

Example 2.
%let y=%str(left parenthesis ();
%put &y;

Example 3.
%let y=this is a single quote %';
%put &y;

Modifying the word scanner's rules
In Example 2 the word scanner was searching for a right parenthesis for the unmatched left parenthesis. If a symbolic variable is required to contain a delta parenthesis we must have a construct that will define a new rule to the wordscanner. The convention implemented in the SAS macro language is to proceed a rule trigger with a percent sign (%). This modification to the wordscanners rules only takes effect in %STR, %NRSTR, and %QUOTE.

We can modify the examples from the previous page:

Example 1.
%let y=left parenthesis ();
%put &y;

Example 2.
%let y=%str(left parenthesis %();
%put &y;

Example 3.
%let y=this is a single quote %str('%');
%put &y;

The characters that can be used with the percent symbol are:
%' an unmatched single quote (')
%" an unmatched double quote (")
%( an unmatched left parenthesis ((
%) an unmatched right parenthesis ()
% a single percent sign (%)
Examples using %STR

Example 1.
%LET FIRSTWRD=%SCAN(&SYSBUFFR,1,%STR( ));

Example 2.
%LET SEMI=%STR(;);
%IF &SORT=YES %THEN
PROC SORT DATA=_LAST_&SEMI
BY &VARS;

Example 3.
%LET TEXT=CREATE A QUOTED STRING;
%LET QUOTE=%STR(%');
%LET QTSTRING=&QUOTE&TEXT&QUOTE;

%NRSTR function

The %NRSTR (short for No Rescan STRing) is identical to the %STR function in handling semicolons and blanks, but it also removes the significance of the ampersand (&) and the percent (%).

%MACRO TALK;
%LET VAR1=%NRSTR(&MONTH);
%LET VAR2=%NRSTR(&YEAR);
%PUT ENTER THE VARIABLES &VAR1 AND &VAR2;
%INPUT;
%PUT VALUE OF &VAR1=%SCAN(&SYSBUFFR,1);
%PUT VALUE OF &VAR2=%SCAN(&SYSBUFFR,2);
%MEND TALK;

The results from invoking the above macro would be:
9 %TALK
ENTER THE VARIABLES &MONTH AND &YEAR
july 1984
VALUE OF &MONTH=july
VALUE OF &YEAR=1984
12

The local symbol table for macro talk would contain two variables:

variable name value
VAR1 &MONTH
VAR2 &YEAR

Example using %NRSTR

%MACRO TEXT;
CARY
%MEND TEXT;

%MACRO EXAMPLE;
%LET A=%TEXT;
%PUT A IS &A;
%PUT %STR( );
%PUT 3."&A" IS DETERMINED BY "%TEXT";
%MEND EXAMPLE;

Invoke the macro EXAMPLE:
A IS CARY
1.CARY IS DETERMINED BY CARY
2."&A" IS DETERMINED BY "%TEXT"
3."CARY" IS DETERMINED BY "CARY"
4.&A IS DETERMINED BY %TEXT

%NRSTR can be used to defer resolution of symbolic variables and the invocation of macros.

Execution time functions

The macro facility has five functions that handle special characters at execution time:

- %QUOTE(argument)
- %BQUOTE(argument)
- %NRQUOTE(argument)
- %NRBQUOTE(argument)
- %UNQUOTE(argument)

%QUOTE function

The %QUOTE function is the execution time equivalent of the %STR compile time function. It is used when special characters in symbolic variables may change the execution of a macro.

%QUOTE masks the meaning of:

- blanks leading and trailing blanks
- ) ( parentheses
- / * + - = ** mathematical operators
- < > = ~ comparison operators
- LE GE LT GT comparison operators
- NE EQ logical operators
- ; , delimiters

%MACRO COND(FIRST,OPER,SECOND,RESULT);
%IF %QUOTE(&FIRST) &OPER %QUOTE(&SECOND) %THEN &RESULT%STR( );
%MEND;
%COND(&STATE,=,OR,PROC PRINT)
The macro evaluator will see

\%EVAL(\text{NORTH CAROLINA} = \text{OR})

and evaluate it as false. If the %QUOTE function was not used the macro evaluator would see

\%EVAL(\text{NORTH CAROLINA} = \text{OR})

which it would interpret as

\text{NORTH CAROLINA} = \text{null OR missing expression}

Rules for using %QUOTE

A set of rules can be established for the use of the %QUOTE function.

* %QUOTE should be used in all implied %EVALS to ensure intended execution.
  \%IF (implied %EVAL) %THEN ;
  \%DO %WHILE (implied %EVAL) ;
  \%DO %UNTIL (implied %EVAL) ;
  \%DO index=exp %TO exp %BY exp ;

* When passing arguments to another macro
  \%VALID(%QUOTE(&PARM))

* When using an unmatched special character in combination with a symbolic variable
  \%LET X=%QUOTE(%I&PARM%I);

\%BQUOTE function

The %BQUOTE (Blind Quote) function performs the same operation on symbolic variables that %QUOTE does.

\%BQUOTE will hide the meaning of the following characters.

<table>
<thead>
<tr>
<th>Characters</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>blanks</td>
<td>leading and trailing blanks</td>
</tr>
<tr>
<td>)</td>
<td>parentheses</td>
</tr>
<tr>
<td>* + - = **</td>
<td>mathematical operators</td>
</tr>
<tr>
<td>/ &lt; &gt; = \</td>
<td>comparison operators</td>
</tr>
<tr>
<td>; :</td>
<td>delimiters</td>
</tr>
<tr>
<td><code> </code></td>
<td>literal definers</td>
</tr>
</tbody>
</table>

The additional function of %BQUOTE is to manipulate text that may contain unmatched quotes or parentheses.

Example of %BQUOTE

Define a macro that will hide all special characters and remove trailing blanks.

\%MACRO HIDE(VARNAMES);
  \%LET &VARNAMES=
  %QUOTE(%BQUOTE(&&VARNAMES));
  %DO I=%LENGTH(&&VARNAMES) %TO 1 %BY -1;
  %LET CHAR=
    %QUOTE(%BQUOTE(%SUBSTR(&&VARNAMES,I,1)));
  %IF CHAR=%STR( ) %THEN %DO;
    %LET &VARNAMES=
    %QUOTE(%BQUOTE(%SUBSTR(&&VARNAMES,1,I)));
    %&VARNAMES
    %GOTO EMIT;
  %END;
  %END;
  %LET &VARNAMES=
  %&VARNAMES
  %EMIT:
  %MEND HIDE;

We can now use the above macro anywhere we would use a macro variable.

\%MACRO TALK;
  \%LET SYSBUFFR=START;
  \%DO %UNTIL(&SYSBUFFR=%STR( ));
    \%INPUT;
    \%PUT %HIDE(SYSBUFFR);
  %END;
  %MEND TALK;
  %TALK

\%NRQUOTE function

The %NRQUOTE (short for No Rescan QUOTE) will mask the meaning of the following characters and strings:

<table>
<thead>
<tr>
<th>Characters</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>blanks</td>
<td>leading and trailing blanks</td>
</tr>
<tr>
<td>)</td>
<td>parentheses</td>
</tr>
<tr>
<td>* + - = **</td>
<td>mathematical operators</td>
</tr>
<tr>
<td>/ &lt; &gt; = \</td>
<td>comparison operators</td>
</tr>
<tr>
<td>LE GE LT GT Eq Ne</td>
<td>comparison operators</td>
</tr>
<tr>
<td>AND OR NOT</td>
<td>logical operators</td>
</tr>
<tr>
<td>; :</td>
<td>delimiters</td>
</tr>
<tr>
<td><code> </code></td>
<td>literal definers</td>
</tr>
</tbody>
</table>

Macro variable resolution and macro invocation will occur when supplied as parameters to %NRQUOTE.
If a macro variable or macro does not exist, that is the symbol table manager refuses the token; \%NRQUOTE will delta the reference.

\%MACRO BLDJCL;
%LET &SYSBUFFR=%STR( );
%PUT ENTER JCL BELOW;
%PUT TO STOP ENTER A NULL LINE;
%DO WHILE(&SYSBUFFR= );
%INPUT;
%GLOBAL JCL&I;
%LET JCL&I=%NRQUOTE(&SYSBUFFR);
%LET SYSBUFFR=&&&JCL&I;
%LET I=%EVAL(&I+1);
%END;
%MEND BLDJCL;

Invoking the macro %BLDJCL

When the macro %BLDJCL is invoked JCL statements are constructed that will contain hidden macro variables.

\%MACRO BLDJCL;
%LET &SYSBUFFR=%STR( );
%PUT ENTER JCL BELOW;
%PUT TO STOP ENTER A NULL LINE;
%DO WHILE(&SYSBUFFR= );
%INPUT;
%GLOBAL JCL&I;
%LET JCL&I=%NRQUOTE(&SYSBUFFR);
%LET SYSBUFFR=&&&JCL&I;
%LET I=%EVAL(&I+1);
%END;
%MEND BLDJCL;

%BLDJCL
//STEP1 EXEC PGM=SAS
//SLIB DD DSN=&SASDSN,DISP=SHR
//WORK DD UNIT=DISK,DSN=&TEMP,DISP=&DISP
//FT11F001 DD SYSOUT=&LOGOUT
//FT12F001 DD SYSOUT=&LISTOUT
null line
VARIABLE VALUE
|----------------------------------|
|JCL1 //STEP1 EXEC PGM=SAS
|JCL2 //SLIB DD DSN=&SASDSN,DISP=SHR
|JCL3 //WORK DD UNIT=DISK,DISP=&DISP
|JCL4 //FT11F001 DD SYSOUT=&LOGOUT
|JCL5 //FT12F001 DD SYSOUT=&LISTOUT

%NRQUOTE function

The %NRQUOTE (short for No Rescan Blind QUOTE) will mask the meaning of the following characters and strings:

| blanks | leading and trailing blanks |
|        | ( )                          |
|        | parentheses                  |
|        | + - = **                     |
|        | mathematical operators       |
| < > = - | comparison operators        |
|        | comparison operators         |
| LE GE LT GT |                     |
| EQ NE | logical operators           |
| AND OR NOT |                      |
| & % | macro triggers              |
| ; | literal definers            |
Macro variable resolution and macro invocation will occur when they are supplied as parameters to `%NRQUOTE.

Unmatched quotes and parenthesis are allowed and will be converted.

If a macro variable or macro does not exist, that is the symbol table manager refuses the token `%NRQUOTE will delta the reference.

Note that `%NRQUOTE and `%NRBQUOTE are very special application functions. Do not attempt to use them without understanding tokenization and resolution. Both functions will delta the results only after full tokenization and resolution.

%UNQUOTE function

The %UNQUOTE function returns the meaning to all characters or strings that %STR, %NRSTR, %QUOTE, %BQUOTE, %NRQUOTE, and %NRBQUOTE mask.

%UNQUOTE returns the meaning of

blanks leading and trailing blanks
) ( parentheses
* + = ** mathematical operators
< > = comparison operators
LE GE LT GT comparison operators
EQ NE logical operators
| AND OR NOT logical operators
; ; delimiters. Do not
# # literal definers
& % logical operators
$&& $&& symbolic resolution and macro invocation

When the SAS macro language encounters a %UNQUOTE the following occurs:

before invoking

after invoking

function

mask

PROC SORT; A; |-------| | PROC SORT; A | reverse |
|-------| | |
A | delta |
|-------| | |
A+ | function + A+ |
A- | function A- |
A X A+ AY | function & A X A+ AY |
AY | function AY |
|-------| | |
C |-------| | C |
| | | |
O | | O |
F | | P |
R | | R |
S | | S |
T |-------| | T |

The %UNQUOTE function takes the delta characters and returns their macro meaning. The character string returned is RESCANNED by the word scanner and any operation is performed.

%UNQUOTE and the other macro functions.

<table>
<thead>
<tr>
<th>%STR</th>
<th>%NRSTR</th>
<th>%QUOTE</th>
<th>Standard</th>
</tr>
</thead>
<tbody>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>----------</td>
</tr>
</tbody>
</table>

%UNQUOTE returns the meaning to special characters that %STR, %NRSTR, %QUOTE, %BQUOTE, %NRQUOTE, and %NRBQUOTE remove.

Example of using %UNQUOTE

Write a macro that will produce an assignment statement. For simplicity assume that the assignment statement is for a character literal and single quotes are used to define the literal.

PROGRAM RUN UNDER SAS RELEASE 82.4
PROCESSOR USED IN EXAMPLE IBM 3083
NOTE, SAS OPTIONS SPECIFIED ARE,
SYSIN~SYSIN

NOTE, OPTIONS MPRINT;
%MACRO ASSIGN(VAR,VALUE)/STMT;
%IF &VALUE GT Z %THEN
%LET TYPE~NUMERIC;
%ELSE %LET TYPE~CHARACTER;
%IF &TYPE~NUMERIC %THEN %00;
%LET &VAR~&VALUE;
%ENO;
%IF &TYPE~CHARACTER %THEN %DO;
&VAR~%STR(%')&VALUE%STR(%');
%END;
%MEND;

OPTIONS IMPLMAC;
DATA;
ASSIGN X FACILITY
+ X~'FACILITY';

ERROR: 307 309
ERROR: 301

0301: INVALID SYNTAX, OR MISSING INFIX OPERATOR, ', ', ',', OR ')'.
0307: UNRECOGNIZED.
0309: THE EXPRESSION IS INCOMPLETE.
0310: THE VARIABLE FACILITY IS UNINITIALIZED.
Example of using %UNQUOTE

PROGRAM RUN UNDER SAS RELEASE 82.4
PROCESSOR USED IN EXAMPLE IBM 3083
NOTE: SAS OPTIONS SPECIFIED ARE:
   SYSIN=SYSIN NINTERACTIVE NOCENTER

1 OPTIONS MPRINT;
2 %MACRO ASSIGN(VAR,VALUE)/STMT;
3 %IF &VALUE GT Z %THEN
   %LET TYPE=NUMERIC;
4 %ELSE %LET TYPE=CHARACTER;
5 %IF &TYPE=NUMERIC %THEN %DO;
6   &VAR=&VALUE;
7 %END;
8 %IF &TYPE=CHARACTER %THEN %DO;
9   &VAR=\%UNQUOTE(\%STR(\'&VALUE\%STR(\')));
10 %END;
11 %MEND;
12 OPTIONS IMPLMAC;
13 DATA;
14 ASSIGN X FACILITY;
15 X='FACILITY';
16

NOTE: DATA SET WORK.DATA1 HAS
1 OBSERVATION AND 1 VARIABLES.
NOTE: THE DATA STATEMENT USED 412K.

The %UNQUOTE function causes the argument passed to it to be rescanned. The word scanner encounters the apostrophe and begins to collect a token (literal). What was a group of tokens becomes one token when the word scanner finds the ending quote.