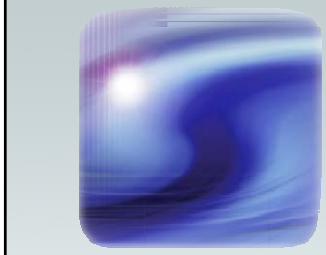


Understanding Why Your Macros Don't Work

M. Michelle Buchecker
SAS Chicago

THE POWER TO KNOW.

Legal Mumbo Jumbo

All material contained in this presentation is copyright 2007 by SAS Institute, Inc.

No piece of this presentation in whole or in part may be distributed electronically or by other means to any person without express written consent by SAS Institute.

In other words, NO SHARING!

Objectives

- When and where you should put quotes
- Describe why you sometimes need extra periods
- Identify why extra ampersands are needed
- Determine the difference between %LET and CALL SYMPUT
- Figuring how and why a macro variable can't be found even though you **know** you created it
- When to use Quoting Functions

Objectives

- **When and where you should put quotes**
- Describe why you sometimes need extra periods
- Identify why extra ampersands are needed
- Determine the difference between %LET and CALL SYMPUT
- Figuring how and why a macro variable can't be found even though you **know** you created it
- When to use Quoting Functions

Once Upon a Time....

Macro variables are used to

- substitute code for code
- create dynamic maintenance-free programs
- have been around since _____.

Version 6

Version 5

Version 4

SAS 82

SAS 76

FOREVER! 

%LET Statement

%LET is a simple way to create a macro variable.

&*macro-variable-name* is a simple way to resolve a macro variable.

```
%LET dwarfs=7;

DATA awards;
     statues=7;
RUN;
```

Quiz

What is the title on the report in the following example?

```
%LET dwarfs=7;

PROC PRINT data=awards;
   title 'There were &dwarfs small statues
awarded in 1939';
RUN;
```

A. There were &dwarfs small statues awarded in 1939

B. There were 7 small statues awarded in 1939

Quote Me On This!

The macro facility does not "peek inside" code with single quotes to resolve macro variables.

You should use double quotes to resolve text that needs quotation marks.

Title "There were &dwarfs small statues awarded in 1939";

Quote Me On This!

The macro facility does not "peek inside" code with single quotes to resolve macro variables.

```
%let year=1939;
Data awards;
  year1=&year;
  year2='&year';
  year3="&year";
Run;
```

year1=1939; → num, 1939
year2='&year'; → char, &year
year3="1939"; → char, 1939

What are the values and types (char or num) of each variable?

Quote Me On This – Part Deux

What if quotes are part of the value?

```
%let year='1939';
Data awards;
  year1=&year;
  year2='&year';
  year3="&year";
Run;
```

year1='1939'; → char, 1939
year2='&year'; → char, &year
year3="'1939'"; → char, '1939'

What are the values and types (char or num) of each variable?

Quote Me On This

Is that a problem?

```
%let year='1939';
```

Write a Title statement that says:

Presented in 1939 by Shirley Temple

(notice there are no quotes around 1939)

```
Title 'Presented in ' &year ' by Shirley Temple';
```


Quote Me On This

Easier to code

```
%let year=1939;
Title "Presented in &year by Shirley Temple";
```

More to Quote on

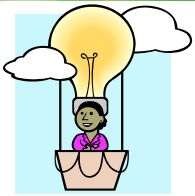
There is a misleading, unwritten rule that states if a quote giving advice comes from someone famous, very old, or Greek, then it must be good advice.

- Bo Bennett

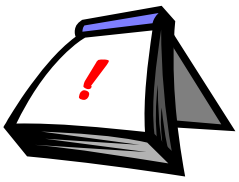
There is a misleading, unwritten rule that states if you put a quote around text, then it must be good for your macro code.

- Michelle Buchecker

Macro Tip

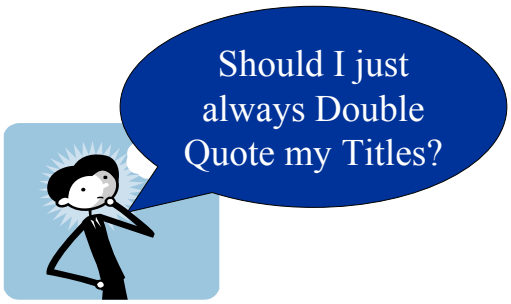


Avoid putting quotes around values in %LET statements



Your Quoteth Runneth Over

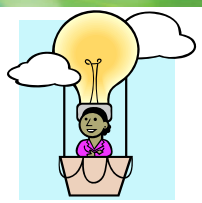
A frequent question is:



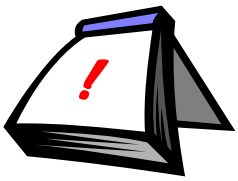
Your Quoteth Runneth Over

```
%let d=Disney;
%let r=Roy;

Desired title:
       R&D created Walt Disney World

Title "R&D created Walt Disney World";

Results in
     RDisney created Walt Disney World
```

Macro Tip



Only use double quotes when necessary.



Objectives

- When and where you should put quotes
- **Describe why you sometimes need extra periods**
- Identify why extra ampersands are needed
- Determine the difference between %LET and CALL SYMPUT
- Figuring how and why a macro variable can't be found even though you **know** you created it
- When to use Quoting Functions

One Dot, Two Dot, Red Dot, Blue Dot

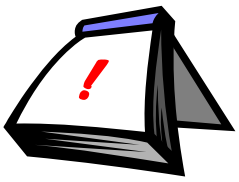
Remember the purpose of the Macro Facility is to do code or text substitution.

For example:

```
Libname company 'c:\my data';
Proc print data=company.MAR2007;
   var revenuecanada revenueus;
Run;
Proc means data=company.MAR2007;
   var revenuecanada revenueus;
Run;
```


Macro Tip

Hardcode everything first and make sure it works!



One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
Libname company 'c:\my data';
Proc print data=company.MAR&year;
   var revenuecanada revenueus;
Run;
Proc means data=company.MAR&year;
   var revenuecanada revenueus;
Run;
```


One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
Libname company 'c:\my data';
Proc print data=company.&month&year;
   var revenuecanada revenueus;
Run;
Proc means data=company.&month&year;
   var revenuecanada revenueus;
Run;
```

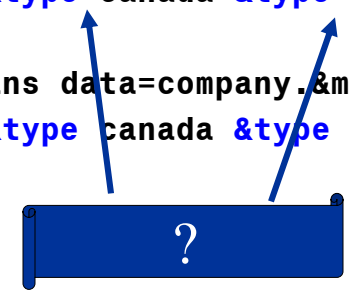
One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
%let type=revenue;
Libname company 'c:\my data';
Proc print data=company.&month&year;
   var &typecanada &typeus;
Run;
Proc means dat                    ear;
   var &typeca
Run;
```

Apparent Symbolic Reference Not Resolved

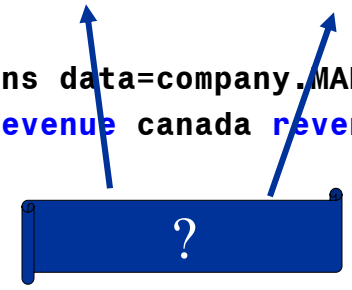
One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
%let type=revenue;
Libname company 'c:\my data';
Proc print data=company.&month&year;
   var &type canada &type us;
Run;
Proc means data=company.&month&year;
   var &type canada &type us;
Run;
```



One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
%let type=revenue;
Libname company 'c:\my data';
Proc print data=company.MAR2007;
   var revenue canada revenue us;
Run;
Proc means data=company.MAR2007;
   var revenue canada revenue us;
Run;
```

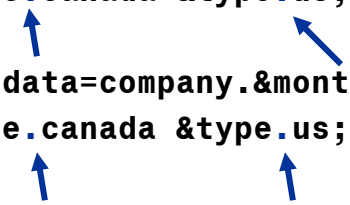


RULE!

Whenever SAS encounters a period after a macro variable reference, the period is treated as a way to end the macro variable and then the period is thrown away.

One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
%let type=revenue;
Libname company 'c:\my data';
Proc print data=company.&month&year;
   var &type.canada &type.us;
Run;
Proc means data=company.&month&year;
   var &type.canada &type.us;
Run;
```

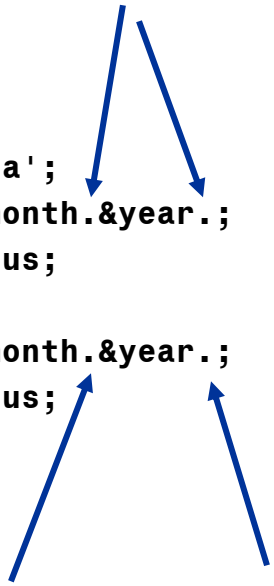


One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
%let type=revenue;
Libname company 'c:\my data';
Proc print data=company.MAR2007;
   var revenuecanada revenueus;
Run;
Proc means data=company.MAR2007;
   var revenuecanada revenueus;
Run;
```

QUIZ! Is this OK???

```
%let year=2007;
%let month=MAR;
%let type=revenue;
Libname company 'c:\my data';
Proc print data=company.&month.&year.;
   var &type.canada &type.us;
Run;
Proc means data=company.&month.&year.;
   var &type.canada &type.us;
Run;
```



Reprise

Whenever SAS encounters a period after a macro variable reference, the period is treated as a way to end the macro variable and then the period is thrown away.

School of Thought

Some SAS programmers like to always put periods after **all** macro variable references.

Some SAS programmers only like to put periods when they are needed.

One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
%let type=revenue;
%let libinfo=company;
Libname &libinfo 'c:\my data';
Proc print data= &libinfo.&month&year;
   var &type.canada &type.us;
Run;
Proc means data= &libinfo.&month&year;
   var &type.canada &type.us;
Run;
```

One Dot, Two Dot, Red Dot, Blue Dot

```
%let year=2007;
%let month=MAR;
%let type=revenue;
%let libinfo=company;
Libname company 'c:\my data';
Proc print data= companyMAR2007;
   var revenuecanada revenueus;
Run;
Proc means data= companyMAR2007;
   var revenuecanada revenueus;
Run;
```

Good

Not Good

Attempt #1

```
%let year=2007;
%let month=MAR;
%let type=revenue;
%let libinfo=company.;
Libname &libinfo 'c:\my data';
Proc print data= &libinfo.&month&year;
   var &type.canada &type.us;
Run;
Proc means data= &libinfo.&month&year;
   var &type.canada &type.us;
Run;
```

Attempt #1

```
%let year=2007;
%let month=MAR;
%let type=revenue;
%let libinfo=company.;
Libname company. 'c:\my data';
Proc print data= company.MAR2007;
   var revenuecanada revenueus;
Run;
Proc means data= company.MAR2007;
   var revenuecanada revenueus;
Run;
```

Not Good

Good

Attempt #2

```
%let year=2007;
%let month=MAR;
%let type=revenue;
%let libinfo=company;
Libname &libinfo 'c:\my data';
Proc print data= &libinfo..&month&year;
   var &type.canada &type.us;
Run;
Proc means data= &libinfo..&month&year;
   var &type.canada &type.us;
Run;
```

If one dot's not enough, use TWO!

Attempt #2

```
%let year=2007;
%let month=MAR;
%let type=revenue;
%let libinfo=company;
Libname company 'c:\my data';
Proc print data= company.MAR2007;
   var revenuecanada revenueus;
Run;
Proc means data= company.MAR2007;
   var revenuecanada revenueus;
Run;
```

Good

Also Good

Reprise (Again!)

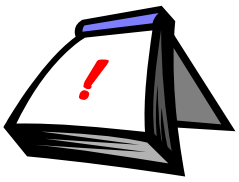
Whenever SAS encounters a period after a macro variable reference, the period is treated as a way to end the macro variable and then the period is thrown away.

Objectives

- When and where you should put quotes
- Describe why you sometimes need extra periods
- **Identify why extra ampersands are needed**
- Determine the difference between %LET and CALL SYMPUT
- Figuring how and why a macro variable can't be found even though you **know** you created it
- When to use Quoting Functions

Macro Tip

Hardcode everything first and make sure it works!



Hardcoded

```
Proc print data=work.all_movies;
   where star='Mickey';
   title 'Mickey is my favorite character';
Run;
```


Common Mistakes

```
%let mouse=Mickey;
Proc print data=work.all_movies;
   where star=&mouse;
   title '&mouse is my favorite character';
Run;
```

What are the two mistakes made?

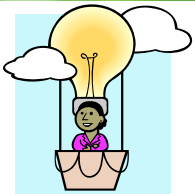
What the Compiler Sees

```
Proc print data=work.all_movies;
   where star=Mickey;
   title '&mouse is my favorite character';
Run;
```

No quotes so SAS thinks this is a variable name.

Remember, the macro facility never peeks inside single quotes.

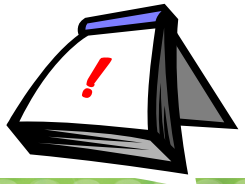
Macro Tip



After the hardcoding works, punch holes in the program and put in your macro variable references.

If you punch a hole inside of a quoted string, change the single quotes to double quotes.

NEVER add or delete quotes from the hardcoded program!

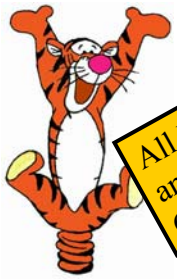


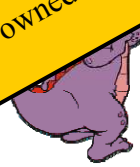
Correct Code

```
%let mouse=Mickey;
Proc print data=work.all_movies;
   where star="&mouse";
   title "&mouse is my favorite character";
Run;
```


Poll!

Who is **your** favorite character?





All Disney names, characters, trademarks, images, parks, and attractions noted are owned by the Walt Disney Company.

Master List of Characters

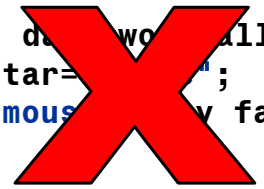
```
%let mouse=Mickey;
%let duck=Donald;
%let princess=Cinderella;
%let tiger=Tigger;
%let dog=Goofy;
%let fairy=Tinkerbell;
%let imagination=Figment;
```


Correct Code

```
%let type=mouse;
Proc print data=work.all_movies;
   where star="?????";
   title "????? is my favorite character";
Run;
```

Attempt #1

```
%let type=mouse;
Proc print data=work.all_movies;
   where star="&type";
   title "&type is my favorite character";
Run;
```



```
Proc print da    wo    ll_movies;
   where star=      ";
   title "mous      y favorite character";
Run;
```


Attempt #2

```
%let type=mouse;
Proc print data=work.all_movies;
   where star="&mouse";
   title "&mouse is my favorite character";
Run;
```

```
Proc print data=work.all_movies;
   where star="Mickey";
   title "Mickey is my favorite character";
Run;
```

OK so far, but no different than before

Attempt #2 Continued

```
%let type=duck;
Proc print data=work.all_mov
   where star="&duck";
   title "&duck is my favori
Run;

Proc print data=work.all_movies;
   where star="Donald";
   title "Donald is my favorite character";
Run;
```

This statement is never even used

TWO changes needed if you want to change the value

Attempt #2 Continued

```
%let type=duck;
Proc print data=work.all_mov
   where star="????";
   title "???? is my favorite character";
Run;
```

Somehow need to reference &type here

And here

Attempt #3

```
%let type=duck;
Proc print data=work.all_movies;
   where star="&type";
   title "&type is my fa
Run;

Proc print da    wo    ll_movies;
   where star=      ;
   title "duck      favorite character";
Run;
```

Oh yeah we tried that back on attempt #1. It still doesn't work.

Attempt #4

```
%let type=duck;
Proc print data=work.all_movies;
   where star="&&type";
   title "&&type is my fa
Run;

Proc print data=work.all_movies;
   where star="duck";
   title "duck is my favorite cha
Run;
```

If one & isn't enough use two!

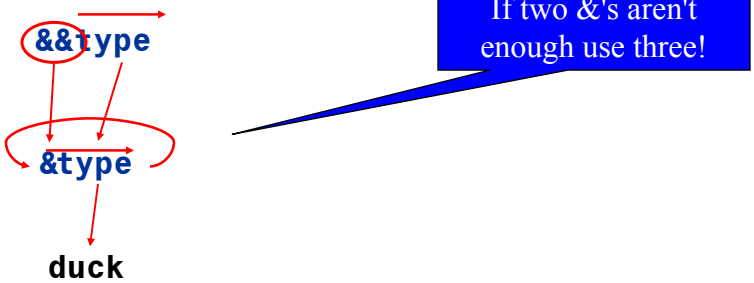
WHAT?!?!?!

RULE!

Whenever SAS encounters two ampersands in a row they resolve down to one ampersand.

The Double Threat

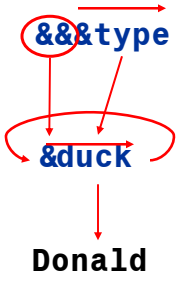
```
%let type=duck;
```



Whenever SAS encounters two ampersands in a row they resolve down to one ampersand.

The Triple Threat

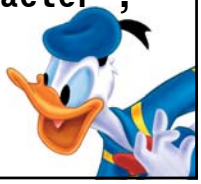
```
%let type=duck;
```



Whenever SAS encounters **two** ampersands in a row they resolve down to one ampersand.

Attempt #5 (and Final)

```
%let type=duck;
Proc print data=work.all_movies;
   where star="&&&type";
   title "&&&type is my favorite character";
Run;

Proc print data=work.all_movies;
   where star="Donald";
   title "Donald is my favorite character";
Run;
```



&&?

So why would you ever need to use only 2 ampersands?

Master List of Mouses

```
%let mouse1=Mickey;
%let mouse2=Minnie;
%let mouse3=Stuart Little;
%let mouse4=Timothy Q. Mouse;
%let mouse5=Gus from Cinderella;
%let mouse6=Miss Bianca;
```

Hardcoded

```
Proc print data=work.all_movies;
   where star='Minnie';
   title 'Minnie is my favorite character';
Run;
```

The Double Threat

What we want:

```
title 'Minnie is my favorite character';
```

What we have:

```
%let num=2;
%let mouse2=Minnie;
```

Is there any macro variable that gives us all or part of **`Minnie is my favorite character`**?

The Double Threat

Punch a hole, and change single quotes to double quotes:

```
title "&mouse2 is my favorite character";
```


The Double Threat

Currently:

```
title "&mouse2 is my favorite character";
```

What we have:

```
%let num=2;
%let mouse2=Minnie;
```

Is there any macro variable that gives us all or part of **mouse2**?

The Double Threat

Currently:

```
title "&mouse2 is my favorite character";
```

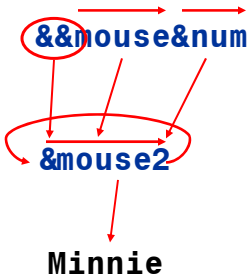
Punch a hole, and change single ampersand to double ampersand :

```
title "&&mouse&num is my favorite character";
```

Is there any macro variable that gives us all or part of **num**?

The Double Threat

```
%let mouse2=Minnie;
%let num=2;
```



Whenever SAS encounters two ampersands in a row they resolve down to one ampersand.

&&

```
%let num=2;
Proc print data=work.all_movies;
   where star="&&mouse&num";
   title "&&mouse&num is my favorite
character";
Run;

Proc print data=work.all_movies;
   where star="Minnie";
   title "Minnie is my favorite character";
Run;
```


Let's Get Fancy!

```
%macro mouses;
  %do num=1 %to 6;
    Proc print data=work.all_movies;
      where star="&&mouse&num";
      title "&&mouse&num is my favorite
             character";
    Run;
  %end;
%mend;

%mouses
```

Quiz 1

What is the title on the report in the following example?

```
%let dwarf1=Dopey;
%let dwarf2=Doc;
%let dwarf3=Sneezy;
%let dwarf4=Happy;
%let dwarf5=Bashful;
%let dwarf6=Grumpy;
%let drawf7=Happy;
%LET dwarfs=5;

PROC PRINT data=awards;
   title "&&dwarf&dwarfs was the hardest working dwarf";
RUN;
```

Bashful was the hardest working dwarf

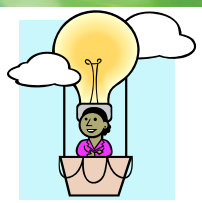
Quiz 2

What is the title on the report in the following example?

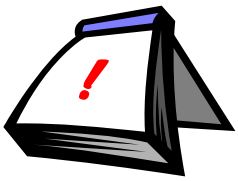
```
%let dwarf1=Dopey;
%let dwarf2=Doc;
%let dwarf3=Sneezy;
%let dwarf4=Happy;
%let dwarf5=Bashful;
%let dwarf6=Grumpy;
%let drawf7=Happy;
%LET dwarfs=8;

PROC PRINT data=awards;
   title "&&dwarf&dwarfs was the hardest working dwarf";
RUN;
```

&dwarf8 was the hardest working dwarf

Macro Tip



Hardcode everything first and make sure it works!



Objectives

- When and where you should put quotes
- Describe why you sometimes need extra periods
- Identify why extra ampersands are needed
- **Determine the difference between %LET and CALL SYMPUT**
- Figuring how and why a macro variable can't be found even though you **know** you created it
- When to use Quoting Functions

Goal

Read in data into a DATA step and create a macro variable based on a condition of the data.

Avoiding Crowds

- Read in 1 month worth of crowd calendar data.
- Crowds are rated from 1 (lowest) to 10 (highest).

- If a crowd level is above a 7, increment a counter.
- If there are more than 15 days of the month where the crowd level is above a 7, create a macro variable that says "Stay Home".
- Otherwise create a macro variable that says "Bring your ears!"

```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
     end;
       %let days=counter;
  end;
run;
```



```
data crowds;
   infile "c:\sugi\macros don't
work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
     end;
       %let days=counter;
  end;
```

```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
     end;
       %let days=counter;
  end;
run;
```



```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7
crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
     end;
       %let days=counter;
  end;
```

```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd
level exceeds a "7" */
   if crowd_level > 7 then
       counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
```



```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15
days where the crowd level is
above a 7 then create a macro
variable that says "stay home".
Otherwise create a macro
variable that says "Bring your
ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
```

```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
     end;
       %let days=counter;
  end;
run;
```



```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
     end;
       %let days=counter;
  end;
```

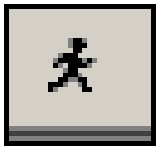
```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your
ears!;
     end;
```



```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then do;
         %let vacation=stay home;
     end;
     else do;
         %let vacation=Bring your ears!;
     end;
       %let days=counter;
  end;
run;
```

```
proc print data=crowds;
title "For this vacation we are going to
       &vacation";
title2 "Because only &days had large
        crowds";
footnote 'April 07 Data retrieved from
          touringplans.com';
run;
```


Run it!



crowds.sas

Problems

Two problems

1. There were 18 days of high crowd levels, yet title said to bring our ears
2. The number of days should have been in title2.

Debug

- Check to see if IF statements are True or False

Back to SAS session

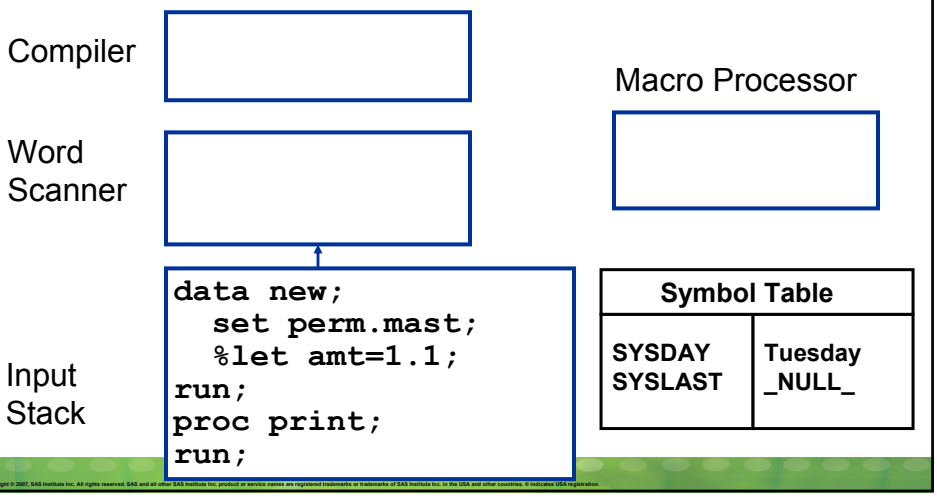
Debug

1. Check to see if IF statements are True or False – IF statements are working properly.

Time to learn some "behind-the-scenes" workings of SAS.

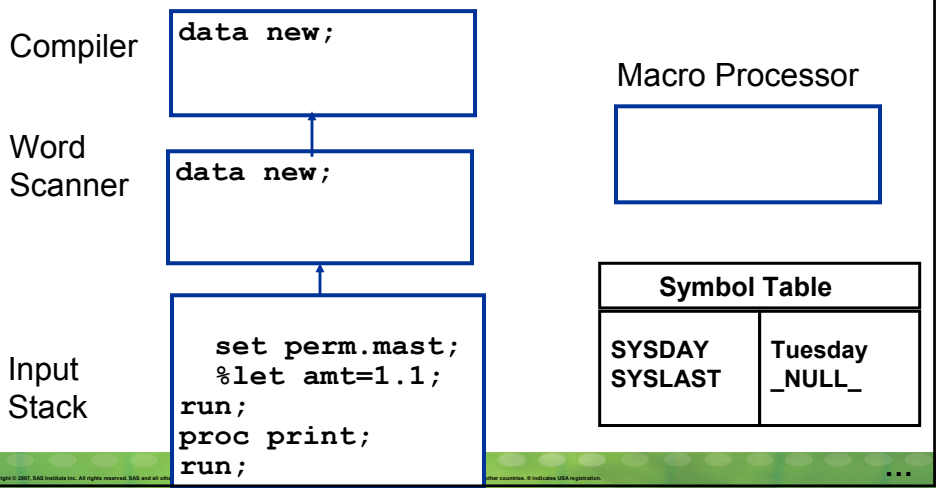
How the Macro Processor Works

Program is put into area of memory called Input Stack.



How the Macro Processor Works

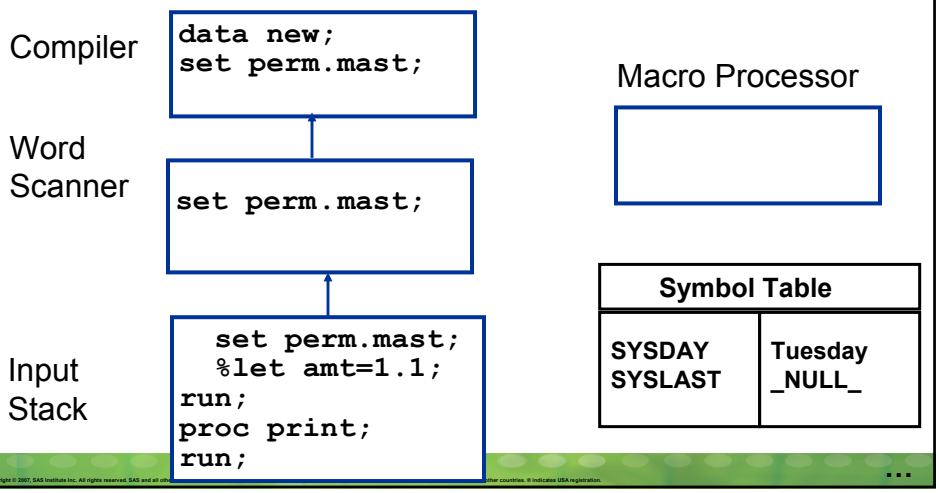
SAS scans 1 word ("token") at a time using the Word Scanner. The Word Scanner usually passes the tokens to the compiler.



...

How the Macro Processor Works

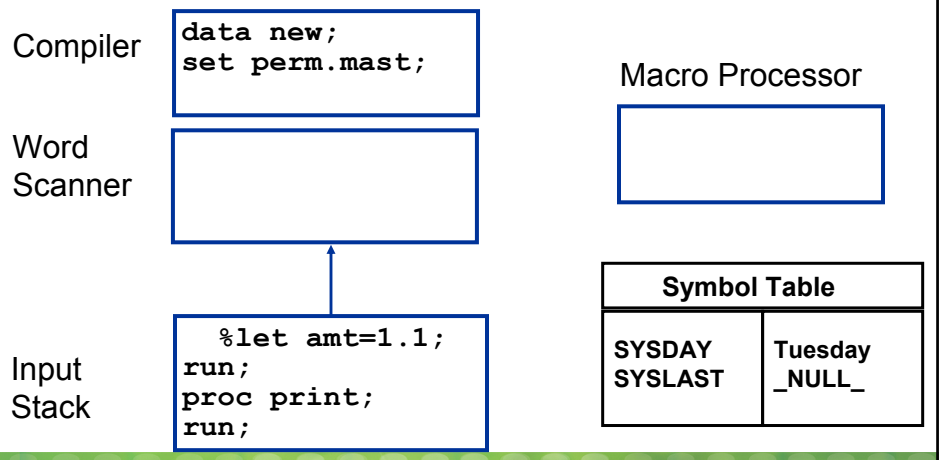
Word scanning continues one statement at a time.



...

How the Macro Processor Works

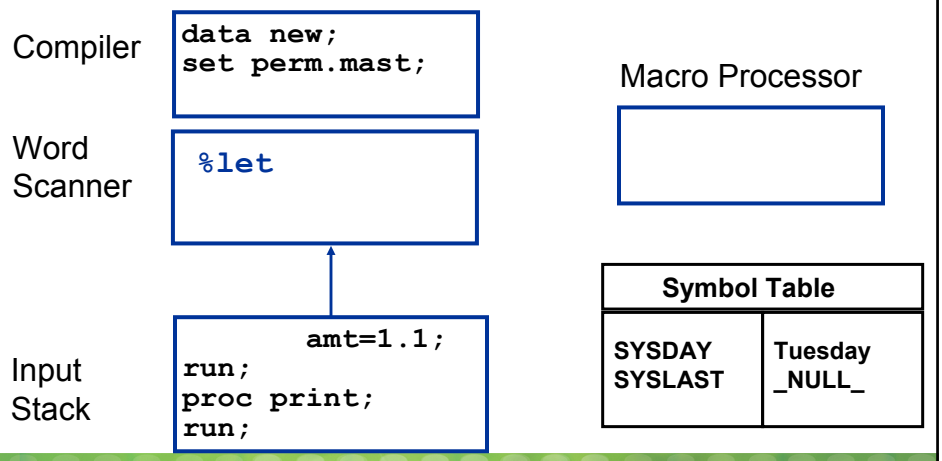
Word scanning continues until a macro trigger is found. A macro trigger is an **&** or **%** followed by a name.



...

How the Macro Processor Works

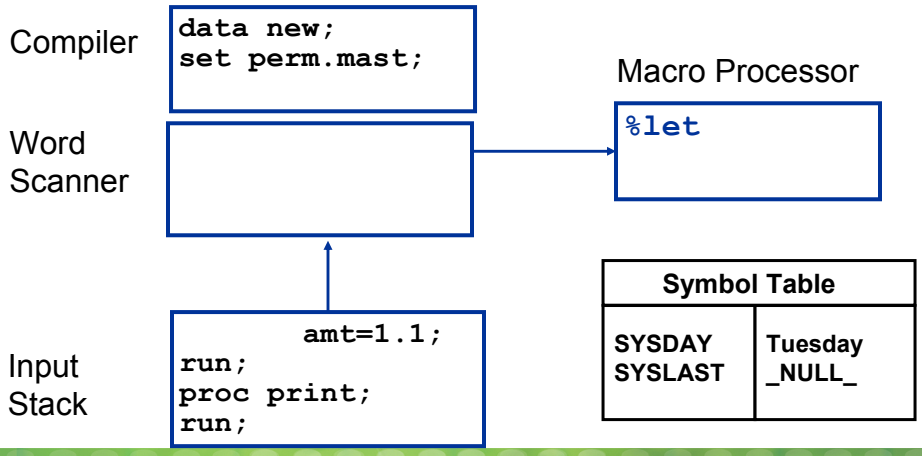
Word scanning continues until a macro trigger is found. A macro trigger is an **&** or **%** followed by a name.



...

How the Macro Processor Works

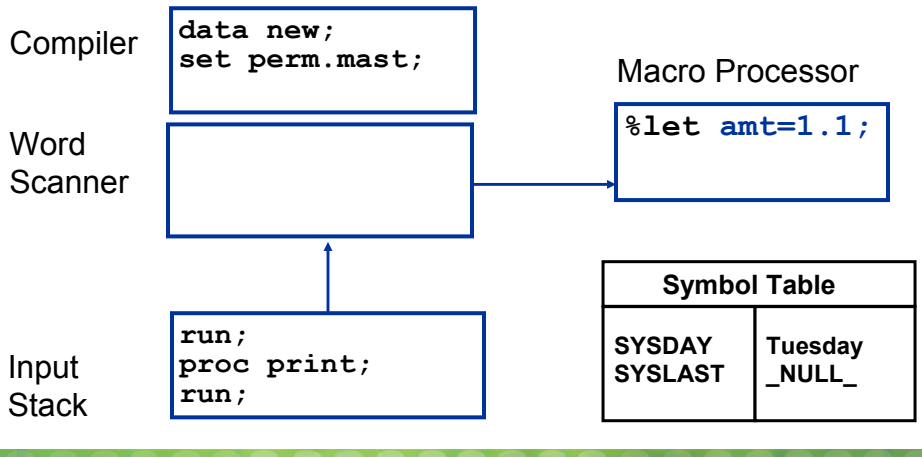
When a macro trigger is encountered, it is passed to the macro processor for evaluation.



...

How the Macro Processor Works

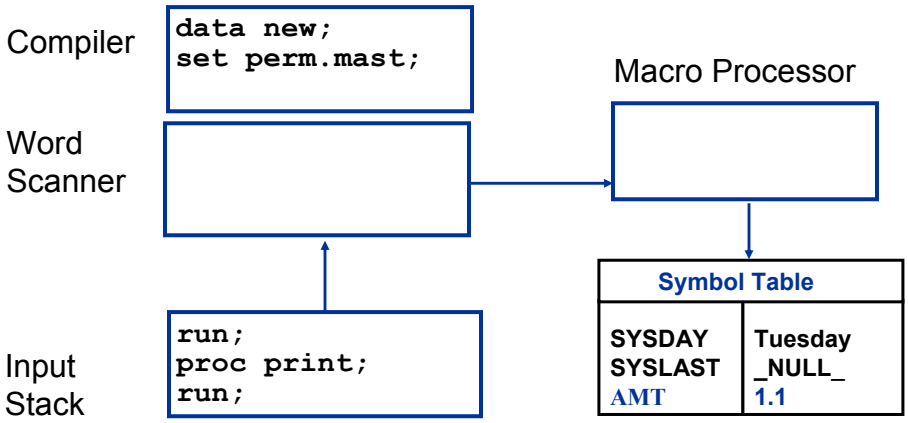
%LET is the keyword of a macro statement that creates a macro variable. The macro processor requests tokens until a semicolon is encountered.



...

How the Macro Processor Works

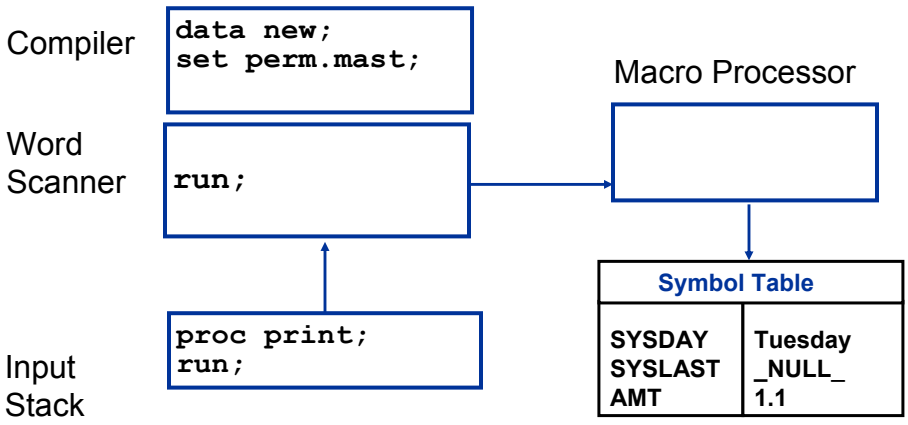
When the %LET statement executes, a macro variable AMT is given the value **1.1** and stored in a memory location called a *symbol table*.



...

How the Macro Processor Works

When a step boundary is recognized, the DATA step compilation phase ends and execution begins.



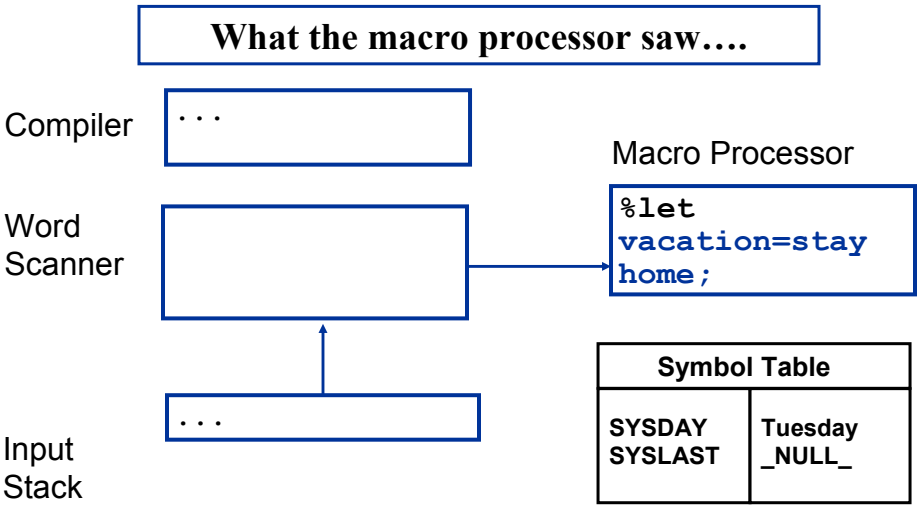
...

Question

Did the Compiler ever see the %LET statement?

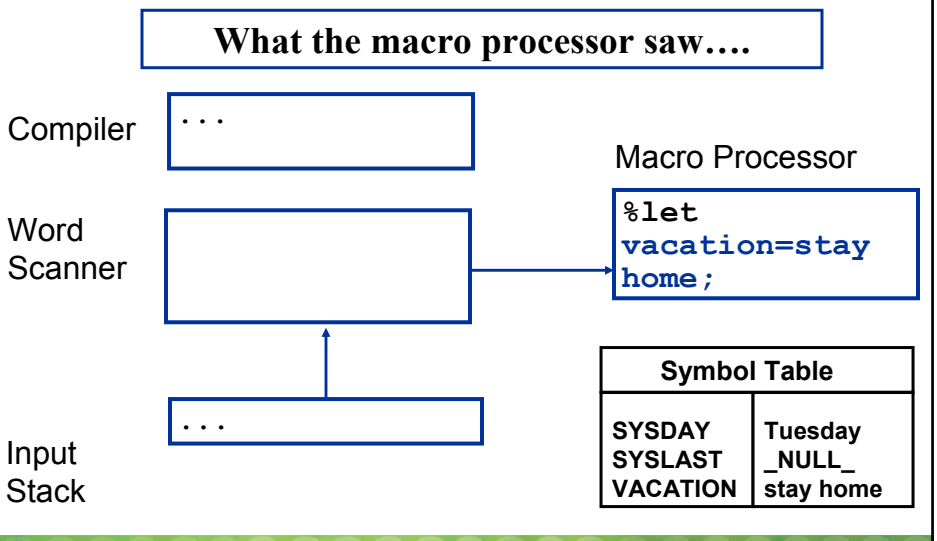
No! %LET like all macro triggers happen during the TOKENIZATION phase.

How the Macro Processor Works



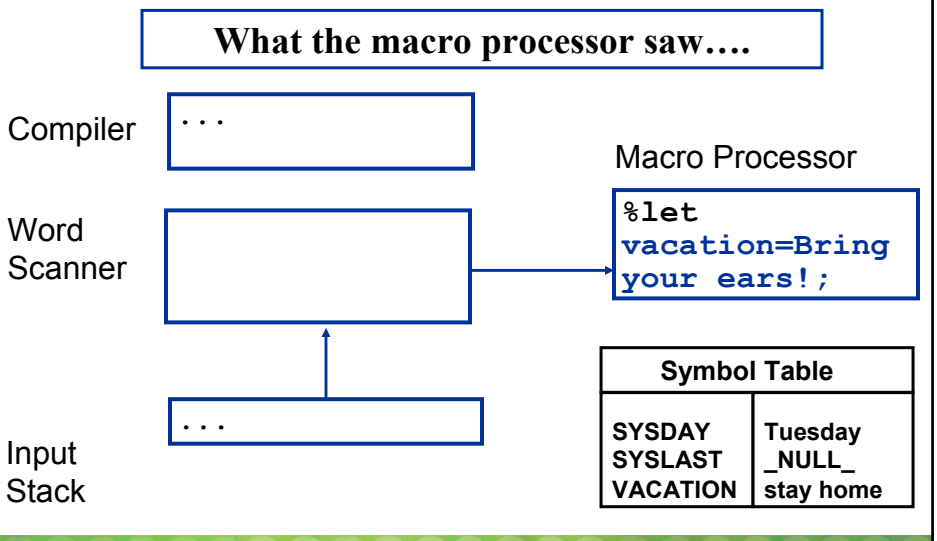
…

How the Macro Processor Works



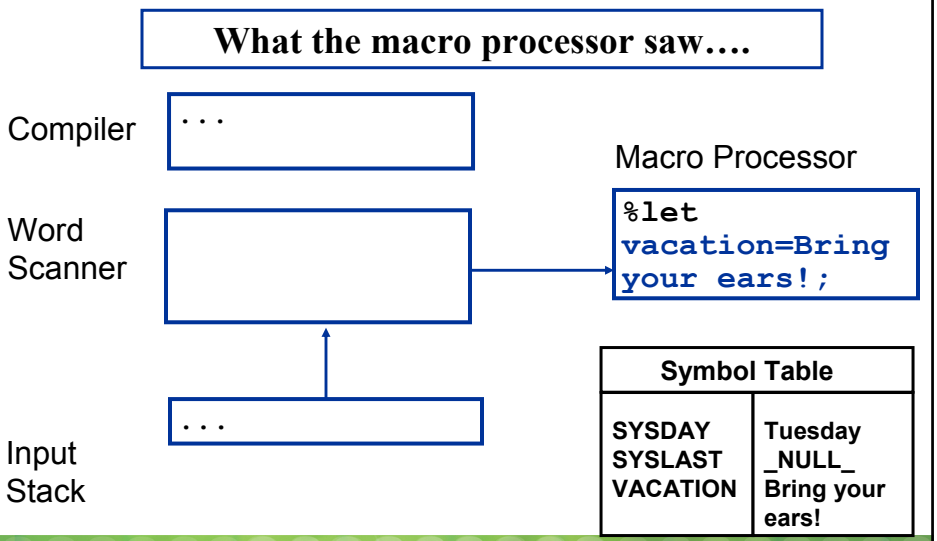
…

How the Macro Processor Works



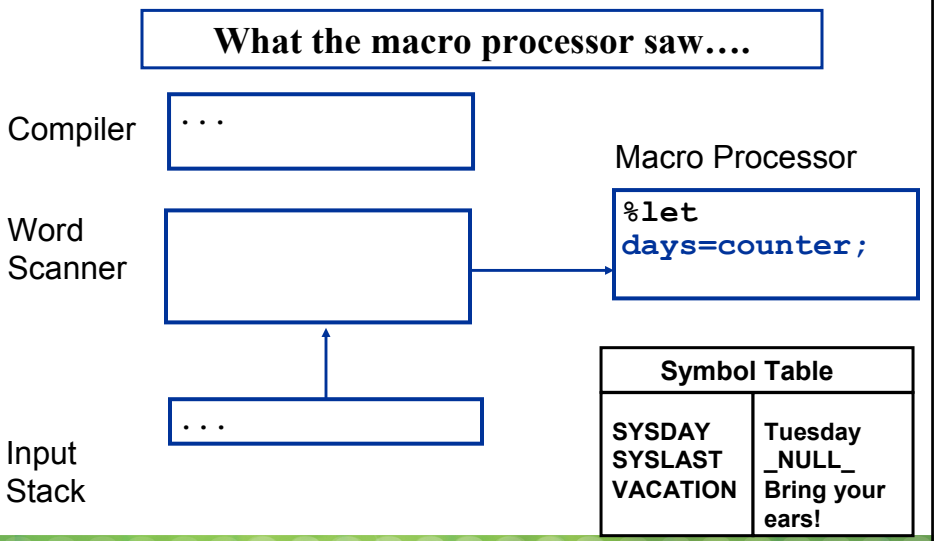
…

How the Macro Processor Works



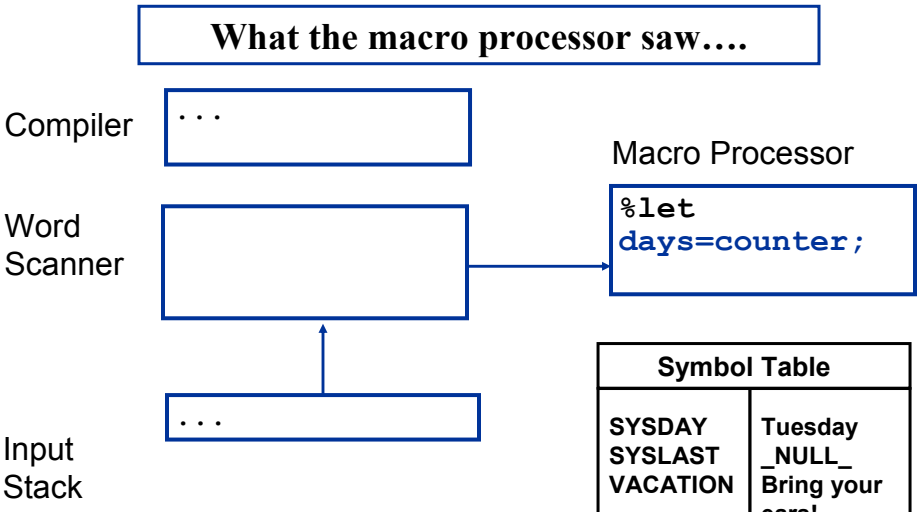
…

How the Macro Processor Works



…

How the Macro Processor Works




```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level
    that s
    variable that says "Bring your ears!" */
   if last then do;
       if counter > 15 then do;
       end;
       else do;
       end;
    end;
run;
```

What the compiler saw….

CALL SYMPUT

Instead of creating a macro variable with %LET, you can use CALL SYMPUT.

The SYMPUT routine

- is an **executable** DATA step statement
- assigns to a macro variable any value available to the DATA step during execution time
- can create macro variables with
  - static values
  - dynamic (data dependent) values
  - dynamic (data dependent) names.

The SYMPUT Routine

You can specify macro-variable or text as a

- Literal `call symput('dog','spot');`
- DATA step variable `call symput('dog',x);`
- DATA step expression `call symput('dog',sum(x,y));`

Dynamically Creating Macro Variables

```
data _null_;
  mickey = 'Mouse';
  goofy = 'Dog';
  call symput('goofy',mickey);
run;
```

What is the macro variable? goofy

What is the value? Mouse

Dynamically Creating Macro Variables

```
data _null_;
  mickey = 'Mouse';
  goofy = 'Dog';
  call symput(goofy,mickey);
run;
```

What is the macro variable? Dog

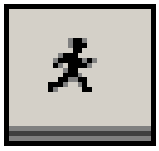
What is the value? Mouse

```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
    variable that says "Bring your ears!" */
  if last then do;
     if counter > 15 then
         call symput('vacation', 'stay home');
     else
       call symput('vacation', 'Bring your ears!');
     call symput('days', counter);
  end;
run;
```



```
data crowds;
  infile "c:\sugi\macros don't work\crowd calendar.dat"
     end=last;
  input dow $ 1-3 day 6-7 crowd_level 18-19;
  /* count number of days crowd level exceeds a "7" */
  if crowd_level > 7 then counter + 1;
  /* if there are more than 15 days where the crowd
    level is above a 7 then create a macro variable
    that says "stay home".  Otherwise create a macro
if counter > 15 then
  call symput('vacation', 'stay home');
else
  call symput('vacation', 'Bring your
ears!');
call symput('days', counter);
run;
```

Run it!



crowds symput.sas

Playing the "WHAT IF..." Game!

Show it!

what if.sas

Playing the "WHAT IF..." Game!

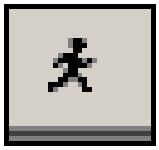
What will the titles on the report be?

A) For this vacation we are going to stay home
Because only 18 days had large crowds

B) For this vacation we are going to &vacation2
Because only &days2 days had large crowds

C) For this vacation we are going to Bring your ears!
Because only 18 days had large crowds

Run it!



what if.sas

What??? WHY????

CALL SYMPUT is a DATA step statement.

The DATA step executes when it hits a step boundary like

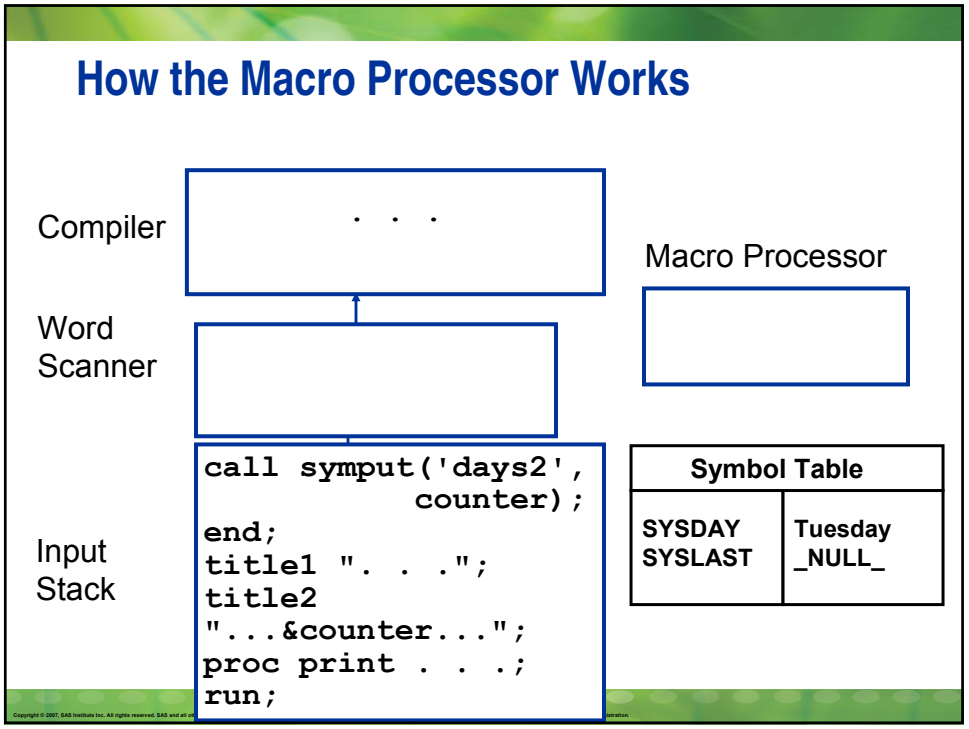
- RUN;
- a new DATA step
- a new PROC step.

What??? WHY????

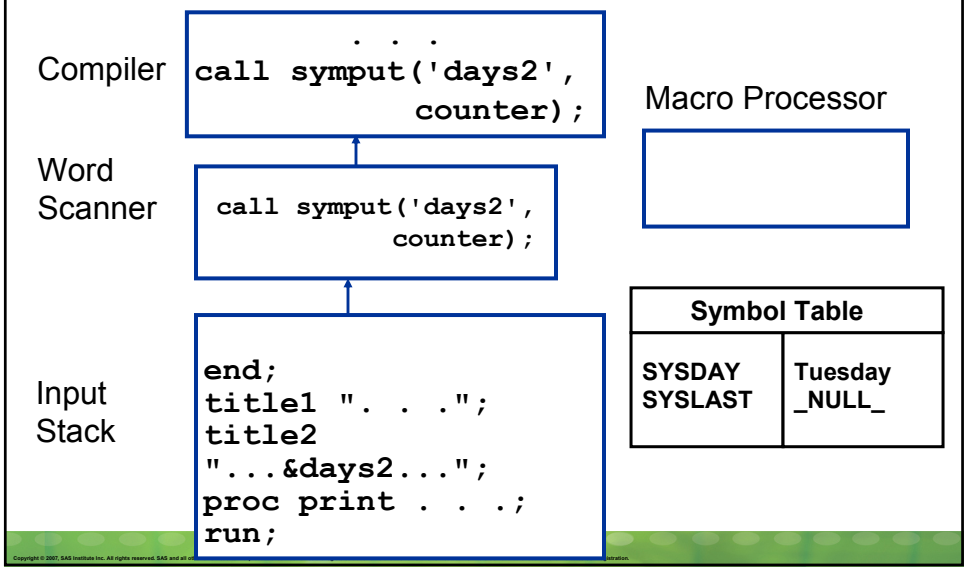
&*macro-variable* is a macro trigger.

It gets resolved during **tokenization**.

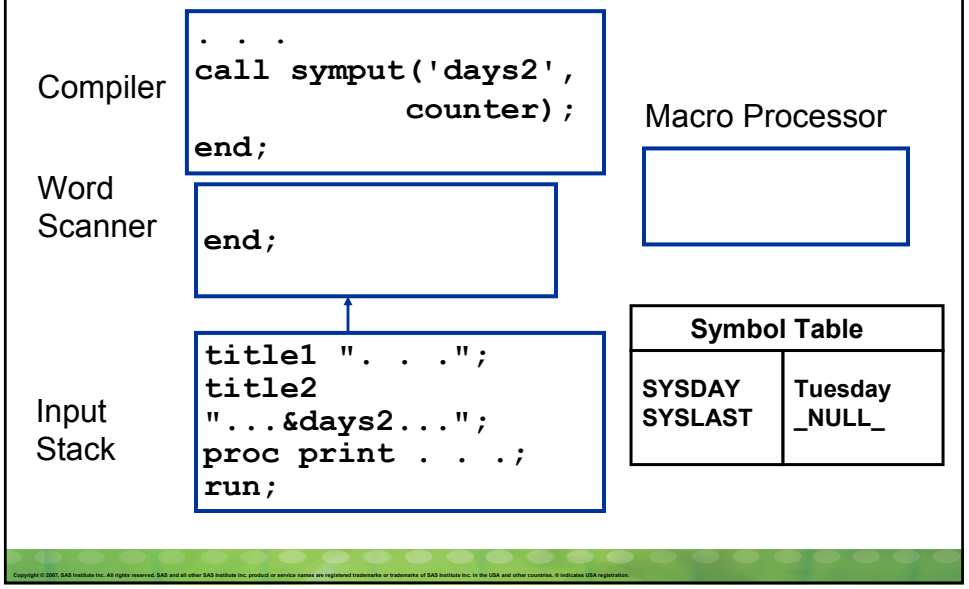
How the Macro Processor Works



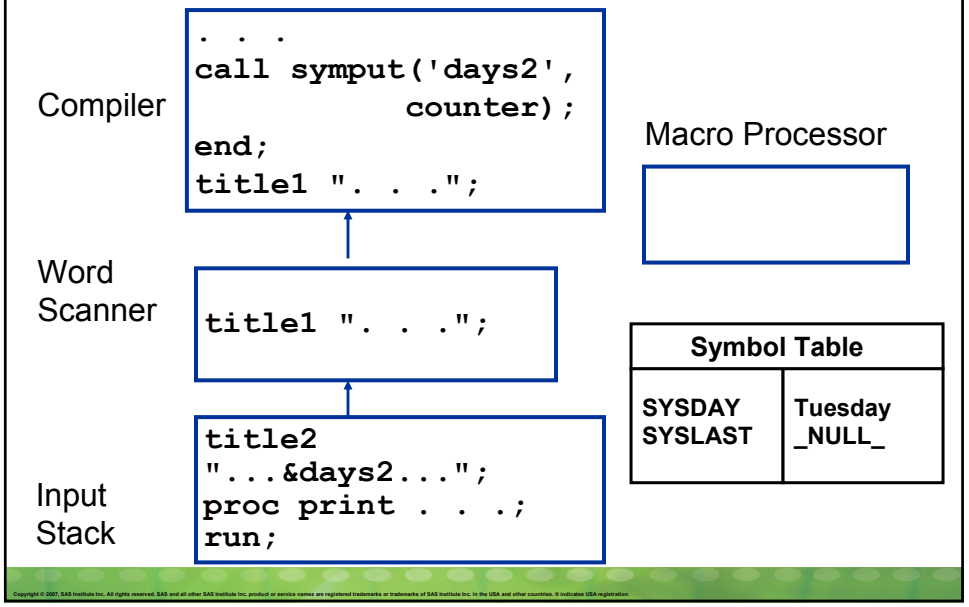
How the Macro Processor Works



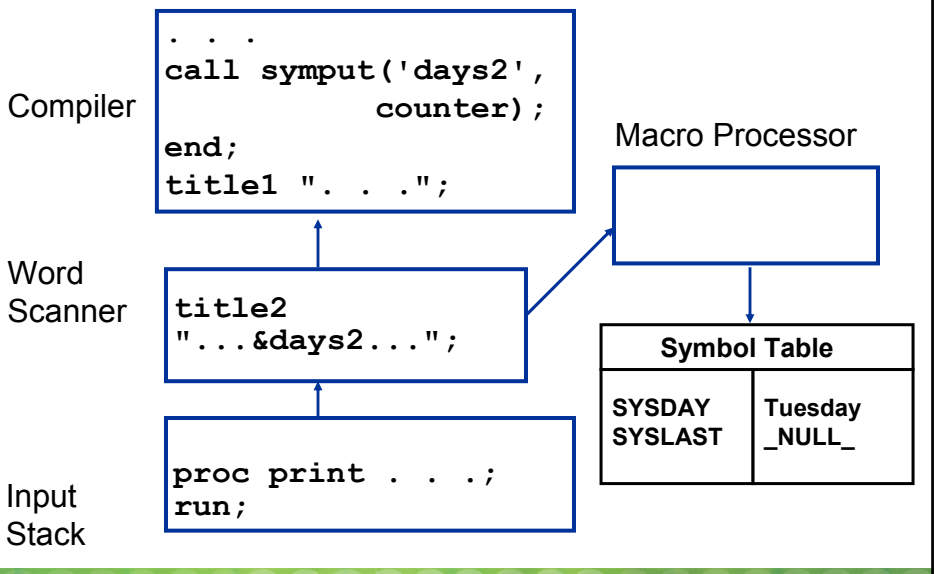
How the Macro Processor Works



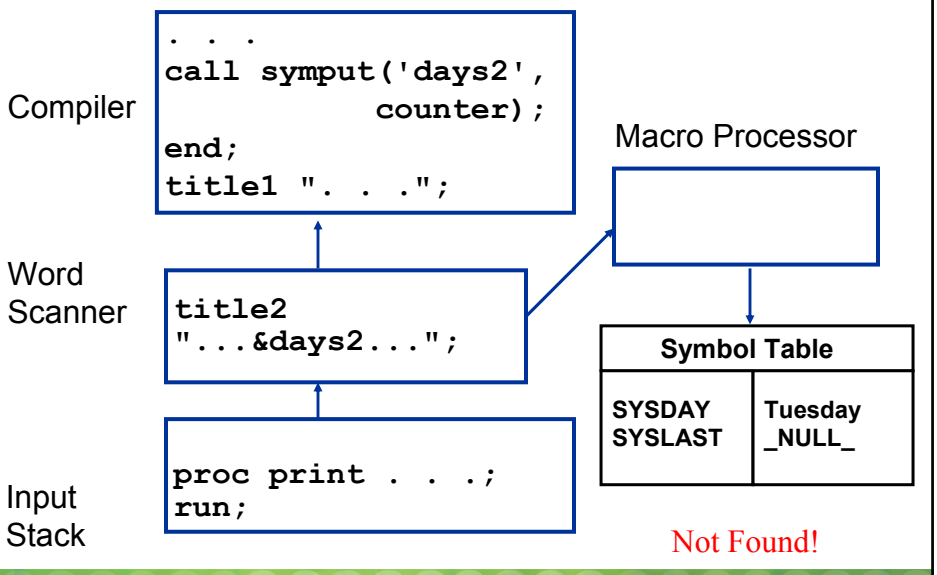
How the Macro Processor Works



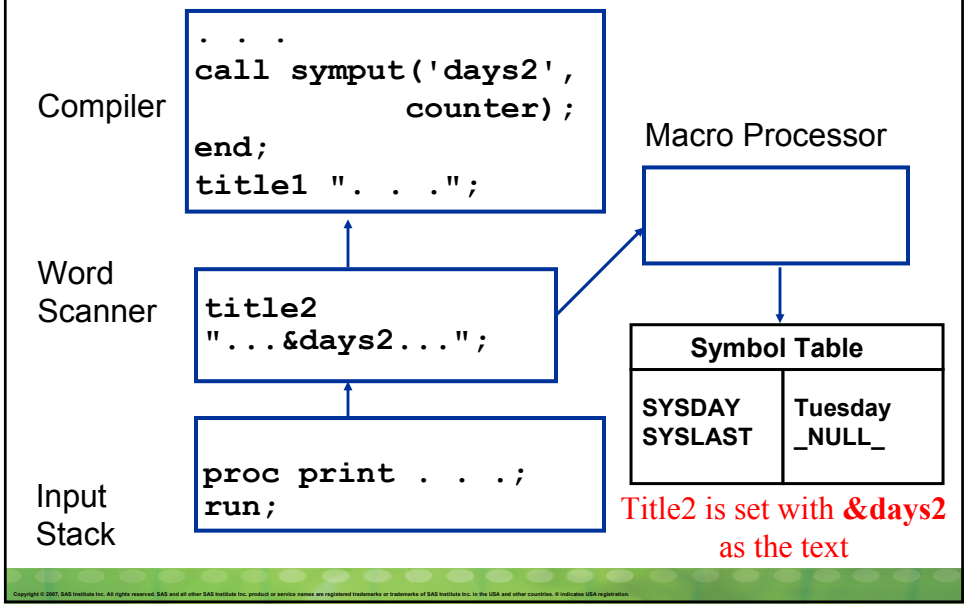
How the Macro Processor Works



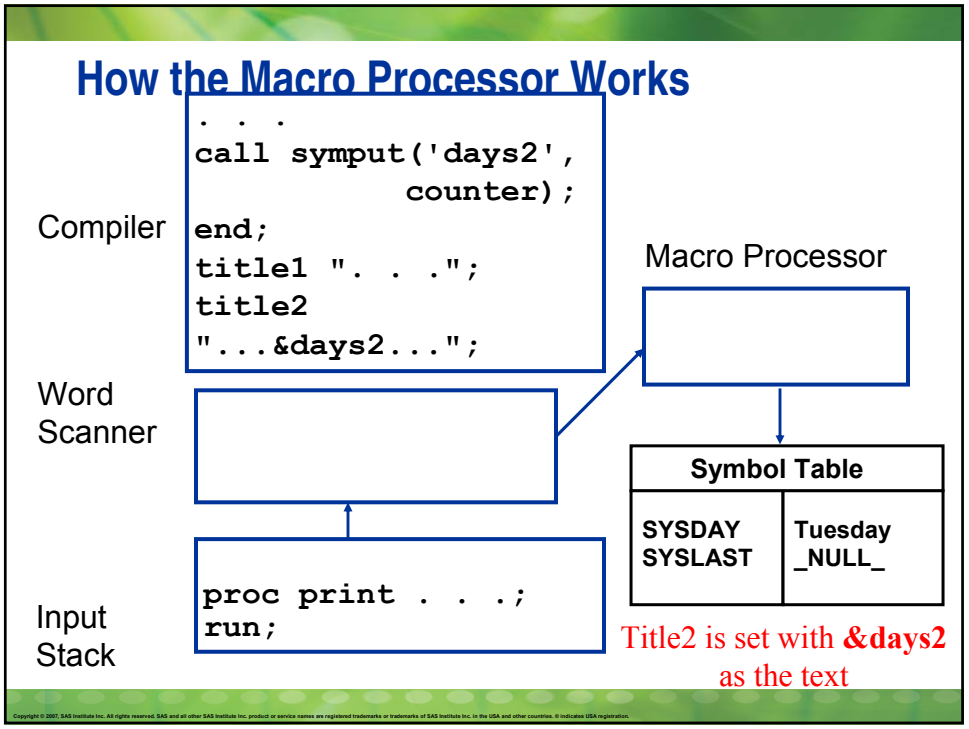
How the Macro Processor Works



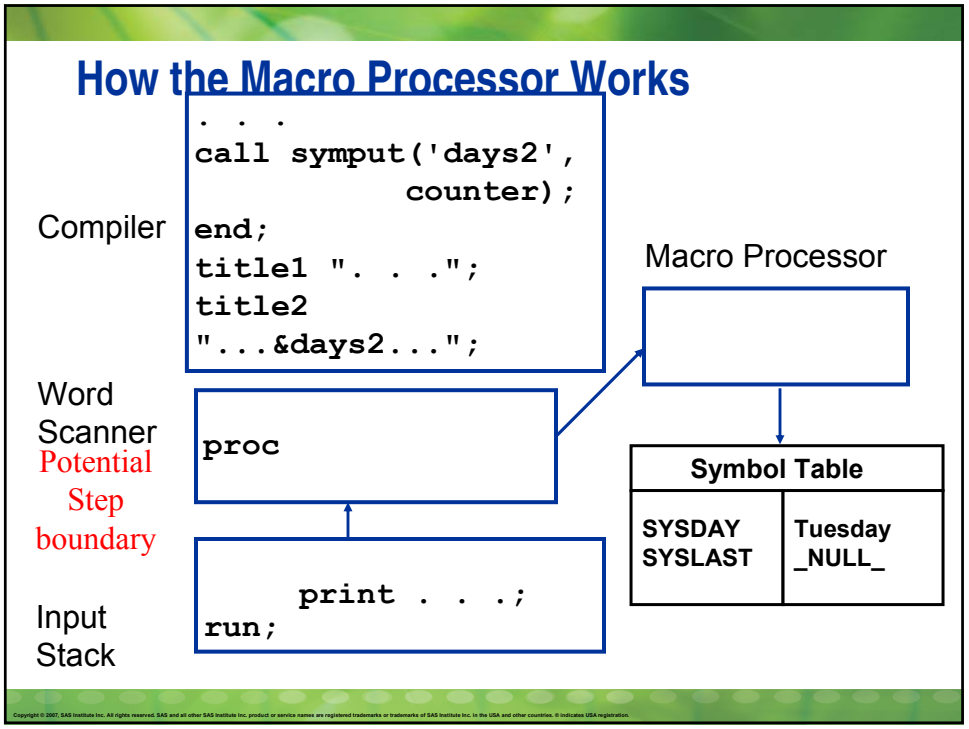
How the Macro Processor Works



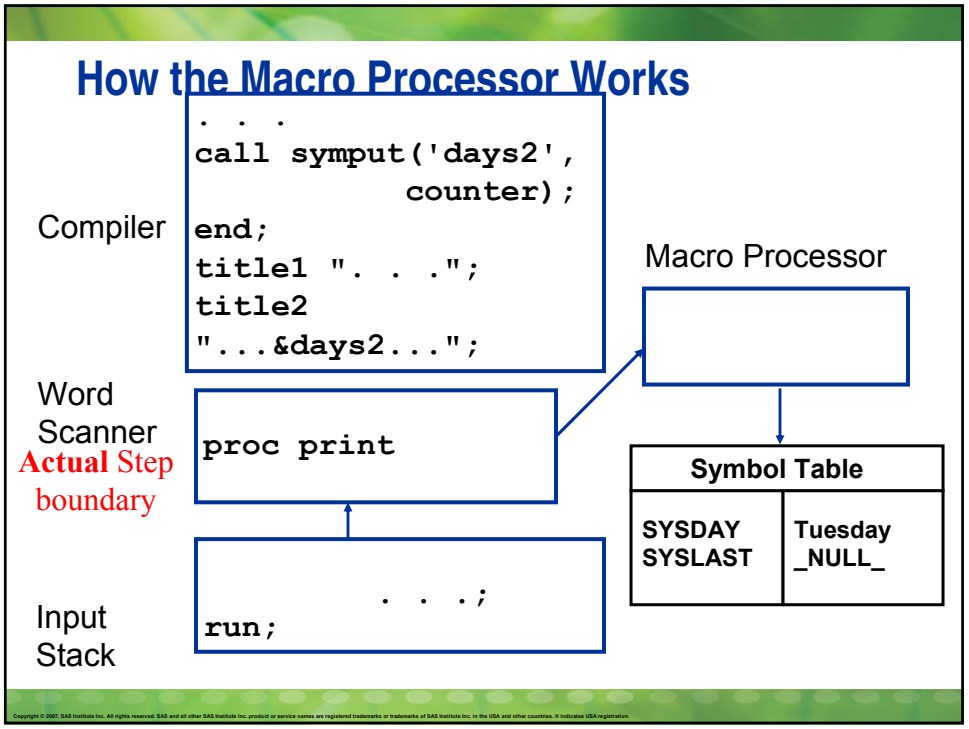
How the Macro Processor Works



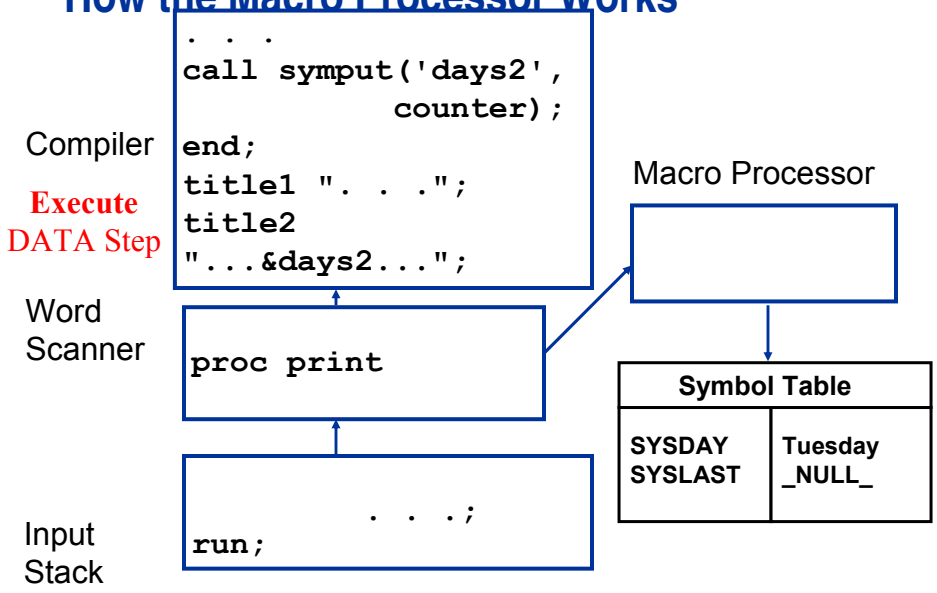
How the Macro Processor Works



How the Macro Processor Works



How the Macro Processor Works



RULE!
Rule of "Slide 130"

When creating a macro variable with CALL SYMPUT, there **MUST** be a step boundary before using it in a macro variable reference (ie. &var)

So….

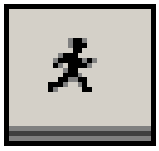
Use a RUN; at the end of the data step!

Objectives

- When and where you should put quotes
- Describe why you sometimes need extra periods
- Identify why extra ampersands are needed
- Determine the difference between %LET and CALL SYMPUT
- **Figuring how and why a macro variable can't be found even though you know you created it**
- When to use Quoting Functions

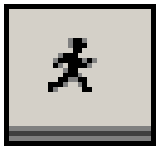

```
%macro create_date;
  /* This is a generic macro that supplies
     today's date in the Month-name day, year
     format and puts that into a macro
     variable  */
  %let today1=%sysfunc(today(), worddate.);
%mend;

proc print data=crowds;
  %create_date
  title "Report created &today1 by me";
run;
```

Run it!



create date let.sas

Run it!



create date sympput.sas

Perplexing

So %LET did not work,
CALL SYMPUT was stable.
Looks like we need to discuss
Global and local symbol table

Global Symbol Table

If you are outside a macro definition all macro variables are placed in the GLOBAL symbol table.

Which means the macro variable is available for the duration of your SAS session.

Local Symbol Table

If you are inside a macro definition all bets are off.

First, SAS sees if there is a macro variable by that name already that it can change the value of.

If there is not a macro variable by that name then the fun begins!

Local Symbol Table and %LET

%LET always puts new macro variables into the LOCAL symbol table.

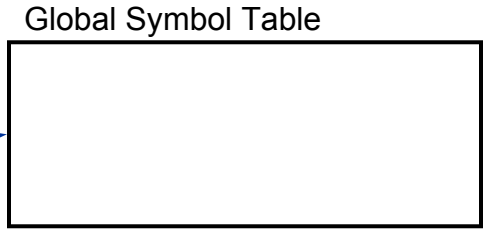
A LOCAL symbol table means the macro variables stored in here are only around for the duration that the macro is executing.

Once the macro finishes execution, BYE-BYE local symbol table and the macro variables in it!

Creating Macro Variables

Outside a Macro Definition

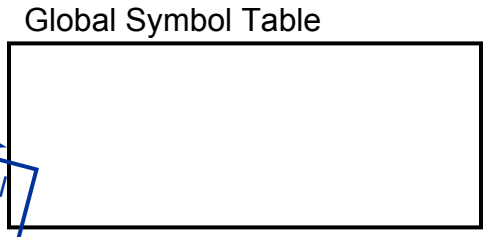
- Call Symput
- INTO clause in PROC SQL
- %LET



Creating Macro Variables

Inside a Macro Definition

- Call Symput
- INTO clause in PROC SQL
- %LET



And NO Local Symbol Table Exists

Creating Macro Variables

Inside a Macro Definition

- Call Symput
- INTO clause in PROC SQL
- %LET

Global Symbol Table

Local Symbol Table

Creating Macro Variables

Inside a Macro Definition

- Call Symput
- INTO clause in PROC SQL
- %LET

IF a Local Symbol Table Exists

Global Symbol Table

Local Symbol Table

Macro Finishes Execution

Inside a Macro Definition

- Call Symput
- INTO clause in PROC SQL
- %LET

Global Symbol Table

Local Symbol Table

Local Symbol Table and %LET

```
%macro create_date;
  /* This is a generic macro that supplies
     today's date in the Month-name day, year
     format and puts that into a macro
     variable  */
  %let today1=%sysfunc(today(), worddate.);
%mend;

proc print data=crowds;
  %create_date
  title "Report created &today1 by me";
run;
```

CALL SYMPUT

```
%macro create_date;
data _null_;
  call symput ('today2',
               put(today(),worddate.));
 run;
%mend;

%create_date
proc print data=crowds;
  title "Report created &today2 by me";
run;
```

CALL SYMPUT

```
%macro create_date;
data _null_;
  call symput ('today3',
               put(today(),worddate.));
 run;
%mend;

proc print data=cr
%create_date
  title "This is the new date &toc
run;
```

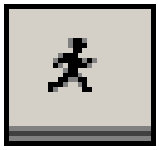
Moved call to macro from **before** PROC PRINT statement to **after**

Changed Title

An Admission

Not intentional that I created this example!!!

Run it!



create date symput 2.sas

What??? AGAIN???? WHY???? WHY????

Think about what the compiler sees:

```
proc print data=crowds;
 %create_date
 data _null_;
   call symput ('today3',
                put(today(),date9.));
 run;
```

Start of new step causes **Execution** of previous step!

Then Macro Variable gets created

What??? AGAIN???? WHY???? WHY????

Think about what the compiler sees:

```
proc print data=crowds;

data _null_;
  call symput ('
 run;

title "This is the new date &today3";
run;
```

Then Title gets set for **next** report

Objectives

- When and where you should put quotes
- Describe why you sometimes need extra periods
- Identify why extra ampersands are needed
- Determine the difference between %LET and CALL SYMPUT
- Figuring how and why a macro variable can't be found even though you know you created it
- **When to use Quoting Functions**

The Crux of It All

What is the SAS Macro Language?

Well Not Quite…

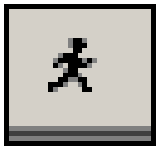
There are certain tokens that have special meaning.

- Blank
- Comma
- &
- %
- others

When is a Comma a Comma?

Sometimes you need to mask the normal meaning of a token and have it just be treated as text.

Run it!



aCOMMAdating.sas

QUIZ!

HINT: Only 1 of these is correct!!!

Given the following program:

```
%let Minnie=Mouse;
data _null_;
  call symput('couple', 'Mickey&Minnie');
run;
```

Which statement correctly displays:

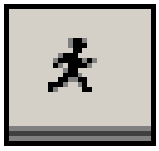
```
couple is Mickey&Minnie
```

a) `%put couple is &couple;`

b) `%put couple is %bquote(&couple);`

c) `%put couple is %superq(couple);`

Run it!



couple.sas

Opposites Attract

What if you wanted to do the complete opposite???

Using %LET, create one macro variable to have a value of **`Mickey&Minnie`**

Create a second macro variable to have a value of **`Mouse`**

Be able to selectively display either

`Mickey&Minnie`

or

`MickeyMouse`

Opposites Attract

Problem #1: Creating the macro variable

What's wrong with:

```
%let Minnie=Mouse;
%let couple=Mickey&Minnie;
```

Always get MickeyMouse

Always Have Protection

We need to protect &Minnie so it doesn't get resolved when the text is stored.

The macro quoting function of **%NRSTR** treats % and & as text (a "quoted" value).

```
%let Minnie=Mouse;
%let couple=%nrstr(Mickey&Minnie);
%put The value is &couple;
```

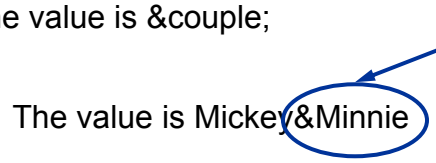
Always Have Protection

We need to protect &Minnie so it doesn't get resolved when the text is stored.

The macro quoting function of **%NRSTR** treats % and & as text (a "quoted" value).

```
%let Minnie=Mouse;
%let couple=%nrstr(Mickey&Minnie);
%put The value is &couple;
```

Now treated as TEXT



The value is Mickey&Minnie

And Make a Resolution

We now **need** for &Minnie to resolve!!!

Remember our ultimate goal is:

The value is MickeyMouse

Quiz!

Fill in the blank:

If we used a quoting function to protect the &, we should use a(n) **unquoting** function to unprotect the &.

Unquoting

The macro unquoting function is **%UNQUOTE** and it treats % and & as macro triggers.

```
%let Minnie=Mouse;
%let couple=%nrstr(Mickey&Minnie);
%put The value is %unquote(&couple);
```

Now macro variable resolves!!!

The value is MickeyMouse

And You Programmed Happily Ever After…

Credits, Questions, Contact Info

Michelle.Buchecker@sas.com