Journey to the center of the earth – Deep understanding of SAS language processing mechanism

Di Chen, SAS Beijing R&D, Beijing, China

ABSTRACT

SAS is a highly flexible and extensible programming language, and a rich library of encapsulated programming procedures. It is designed for data processing and statistical analysis. As we known, SAS is procedure-oriented language which is benefit to underlying operation. However, it will be easy to get the unexpected result although no syntax error during code written if you're not clear the SAS language processing mechanism.

This paper mainly introduce the procedure of processing and resolving in the background of SAS language.

INTRODUCTION

Especially these concepts are helpful for understanding how the SAS compiler works with macro processor together. Including:

1. Word scanner reads the SAS code in the input stack, and tokenizes the SAS program text.
2. Then tokens are passed to the appropriate compiler upon demand:
    - General tokens are passed to the SAS compilers (including DATA Step Compiler, SQL compiler, SCL Compiler and Command Processor).
    - When a macro trigger (% and &) followed immediately by a name token is encountered, the tokens are passed to the macro processor for evaluation.

WHAT IS THE VALUE OF MACRO VARIABLE TODAY?

If you don't understand the SAS language processing mechanism, it is easy to write below code: Use symputx routine to create a macro variable today which value is Friday, and then update the macro variable value using %let statement. However, is the final value of macro variable today really Wednesday?

Wish you can get the correct answer and understand the reason after reading this paper.

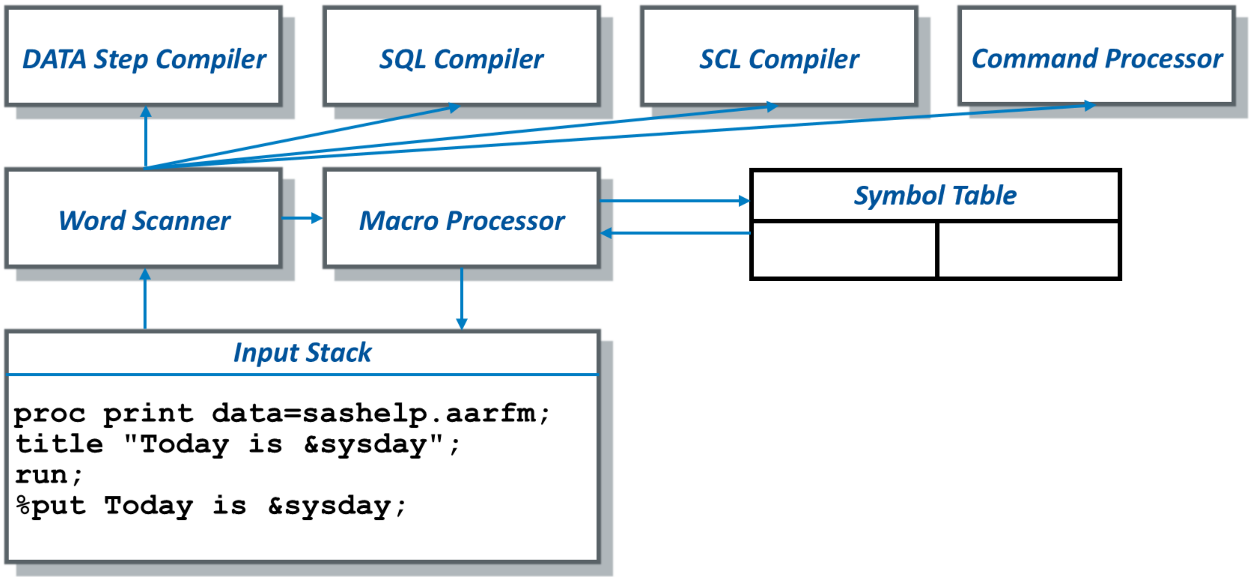
```
data _null_;
call symputx('today','Friday');
%let today=&sysday;
run;
%put Today is &today;
```

Symbol Table	
SYSDAY	Wednesday

Figure 1.1 Example 1

PROCESS FLOW DIAGRAM OF SAS LANGUAGE

When you submit a SAS program, it is copied to a memory area called the input stack. This is true for all code that you submit, such as a DATA step, Proc step, SQL code, or SCL code. The following figure shows the input stack that contains a simple SAS program.



```
proc print data=sashelp.aarfm;
title "Today is &sysday";
run;
%put Today is &sysday;
```

Figure 2.2 Process Flow Diagram of SAS Language

Between the input stack and the compiler(s), SAS programs are tokenized into smaller pieces. The component of SAS known as word scanner which does the tokenization. Once SAS code is in the input stack, word scanner reads the text in the input stack (left-to-right, top-to-bottom), and tokenizes program text. Then tokens are pulled by the appropriate compiler upon demand:

- DATA Step Compiler – DATA steps and PROC steps
- SQL Compiler – Structured Query Language (SQL) code
- SCL Compiler – SAS Component Language (SCL) code
- Command Processor
- Macro Processor – Responsible for handling all SAS macro language elements

HOW SAS RECOGNIZES TOKENS AND HOW THE DIFFERENT PARTS OF SAS LANGUAGE ARE TRANSFERRED TO DIFFERENT COMPILER?

The word scanner recognizes four classes of tokens:

- Literal: a string of characters enclosed in quotation marks.
- Number: digits, date values, time values, and hexadecimal numbers.
- Name: a string of characters beginning with an underscore or letter.
- Special: any character or group of characters that have special meaning to SAS. Examples of special characters include: * / + - ** ; $ ( ) . & % =

A token ends when the word scanner detects a blank, or the beginning of another token. The maximum length of a token is 32,767 characters.

As the program in figure 1.2, the word scanner recognizes the token as the beginning of a PROC step, and triggers the DATA step compiler, which begins to request more tokens. There is no other SAS language element in this example, so other compilers will not be used such as SQL compiler, SCL compiler or command processor. The DATA step compiler pulls tokens from the top of the queue. In most SAS programs with no macro processor activity, all information that the compiler receives comes from the submitted program.

The compiler requests tokens until it receives the end of the step which is called step boundary (in this example, it is the `RUN` statement), or the beginning of a new DATA or PROC step. If the end of the input stack is not a step boundary, the processed statements remain in the compiler.

You can refer to figure 1.3.

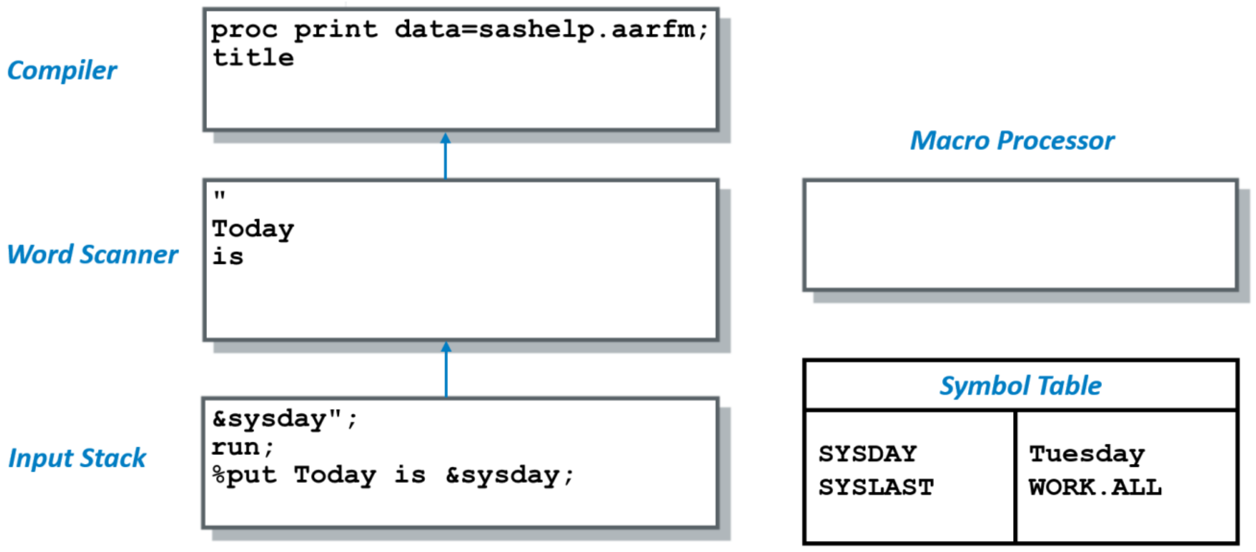


Figure 1.3 Word Scanner Tokenizes Program

When a macro trigger (% or &) followed immediately by a name token is encountered, it alerts the word scanner that the code should be passed to the macro processor for evaluation.

- If the macro trigger begins with %, the macro processor requests additional tokens until a semicolon is encountered, and then it executes the macro statement.
- If the macro trigger begins with &, the macro variable reference triggers the macro processor to search the symbol table for the reference. Then the macro processor resolves the macro variable reference, substituting its value, and pass the resolved reference back to the input stack.

You can refer to figure 1.4.

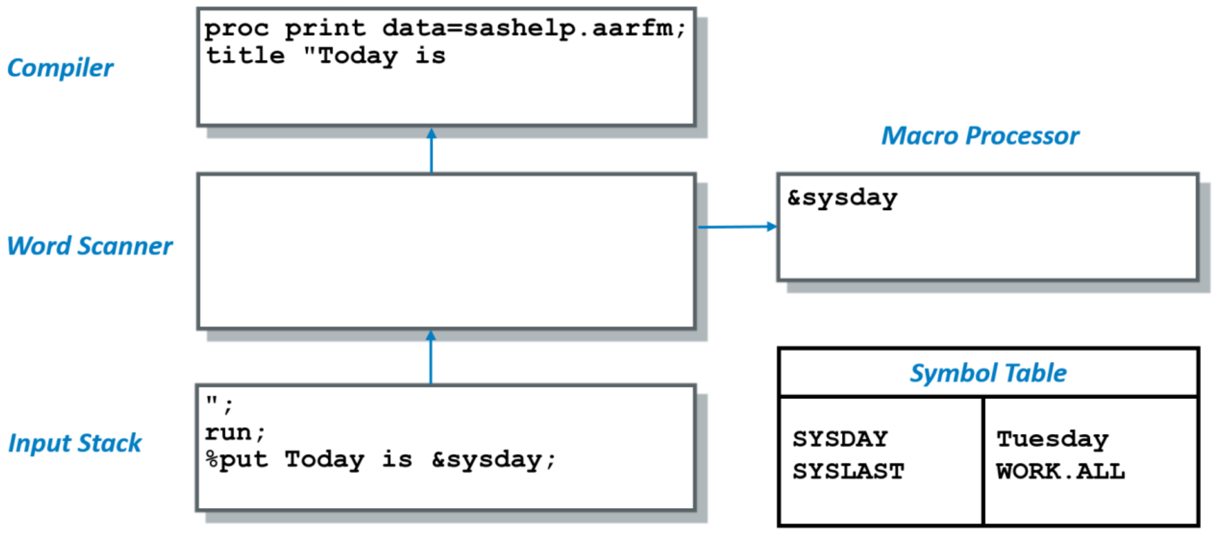


Figure 1.4 Macro Trigger Passed to Macro Processor

Then the macro processor searches the symbol table for resolving the macro variable reference. Automatic macro variables are stored in a memory area called the global symbol table. SAS creates the symbol table at the beginning of a SAS session to hold the values of automatic and global macro variables.

You can refer to figure 1.5.

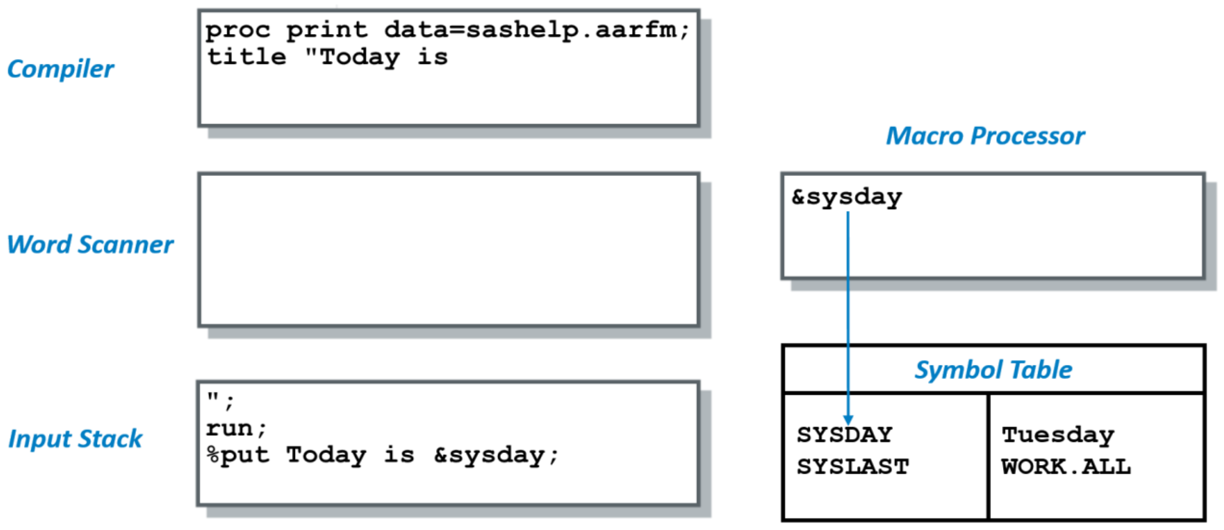


Figure 1.5 Macro Processor Searches Macro Variable Reference

The macro processor resolves the macro variable reference, substituting its value, and pass the resolved reference back to the input stack.

From the time the word scanner triggers the macro processor until that macro processor action is complete, the macro processor controls all activities. When the macro processor is active, no activity occurs in the word scanner or other compilers.

You can refer to figure 1.6.

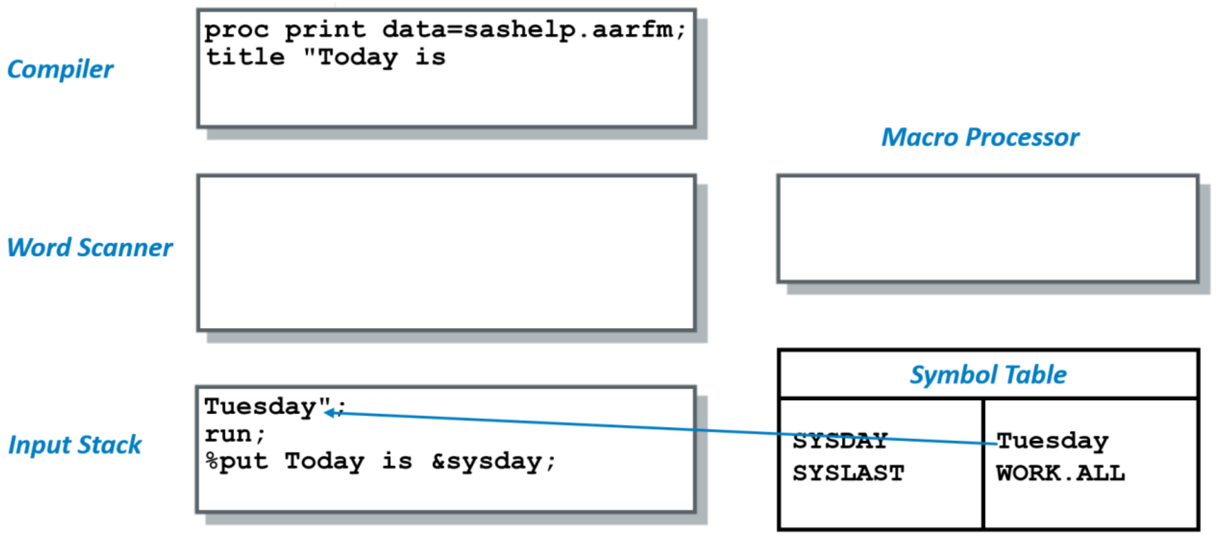


Figure 1.6 Macro Processor Resolves Macro Variable Reference

When the macro processor is finished, the word scanner continue to read the next token and trigger the compiler, which begins to pull tokens from the top of the queue.

You can refer to figure 1.7.

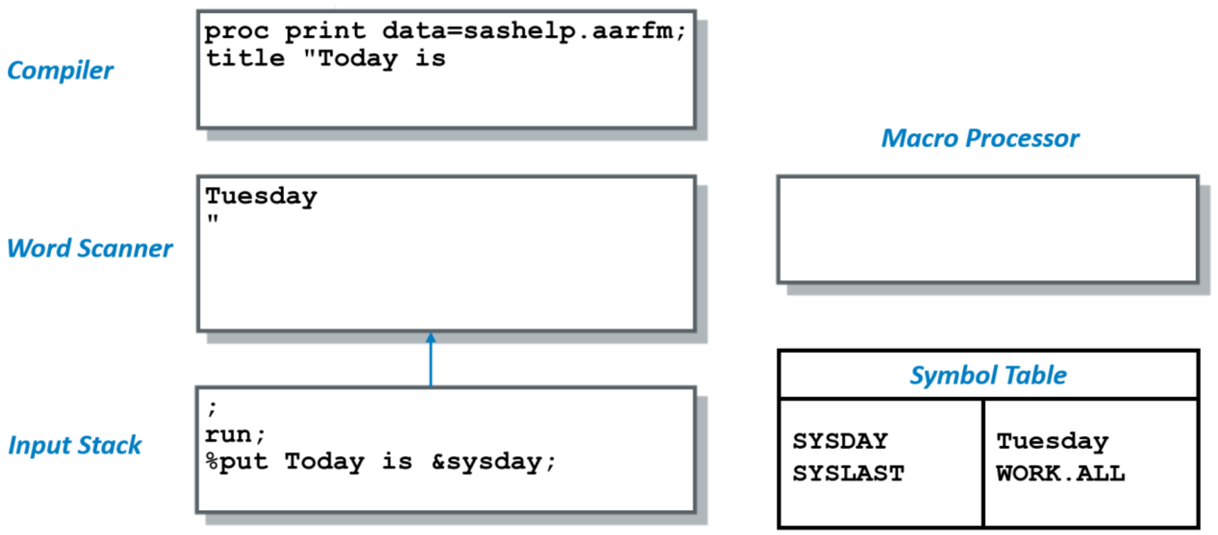


Figure 1.7 Word Scanner Continue to Tokenize Program

When a boundary (`run;`) is reached, the word scanner suspends activity, and the compiler performs a syntax check on the statements. Once the entire step has been compiled without error, it is executed. After finishing execution, SAS then frees the DATA step task. Any macro variables that were created during the program remain in the symbol table.

You can refer to figure 1.8.

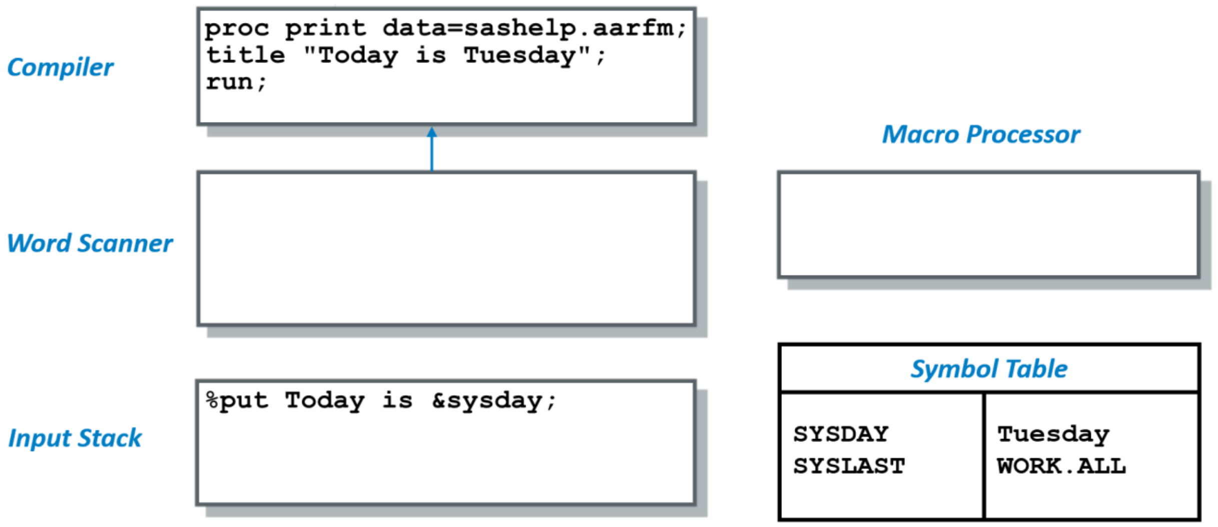


Figure 1.8 Compiler Compile and Execute Program When Boundary reached

The macro processor requests additional tokens for macro trigger % followed immediately by name token until a semicolon is encountered.

You can refer to figure 1.9.

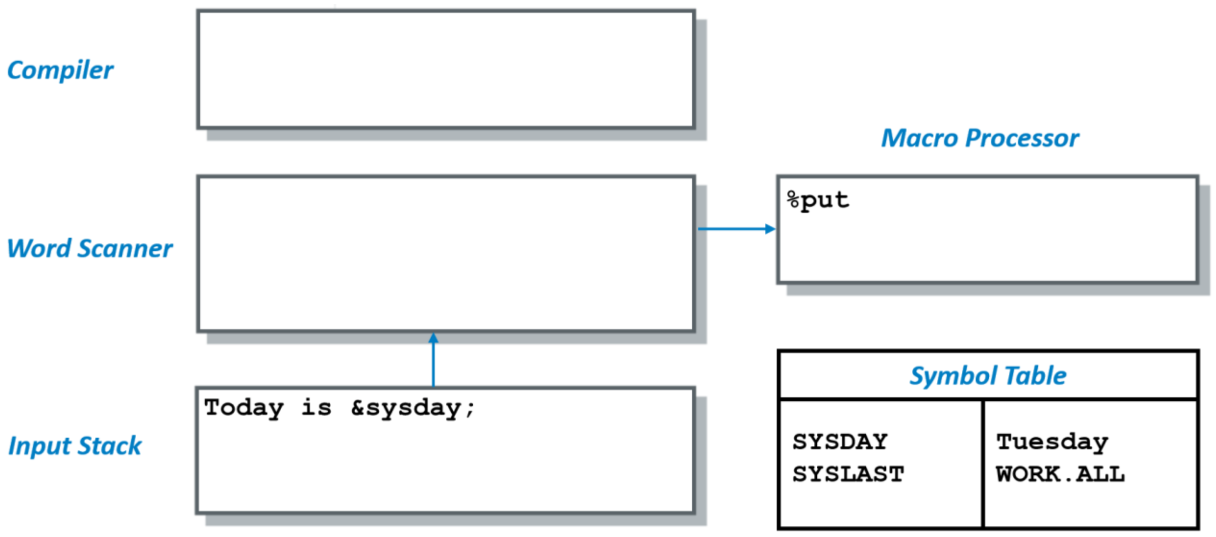


Figure 1.9 Macro Processor Processes Macro statement

The macro processor resolves the macro variable reference and substituting its value. And then executes the macro statement.

You can refer to figure 1.10 and 1.11.

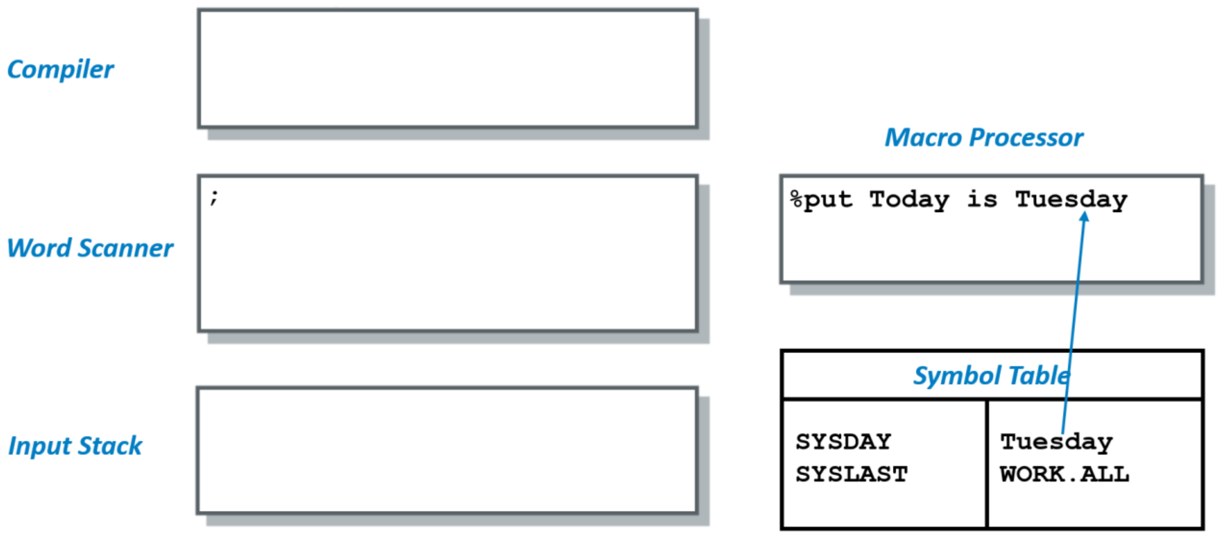


Figure 1.10 Macro Processor Resolves Macro Variable Reference

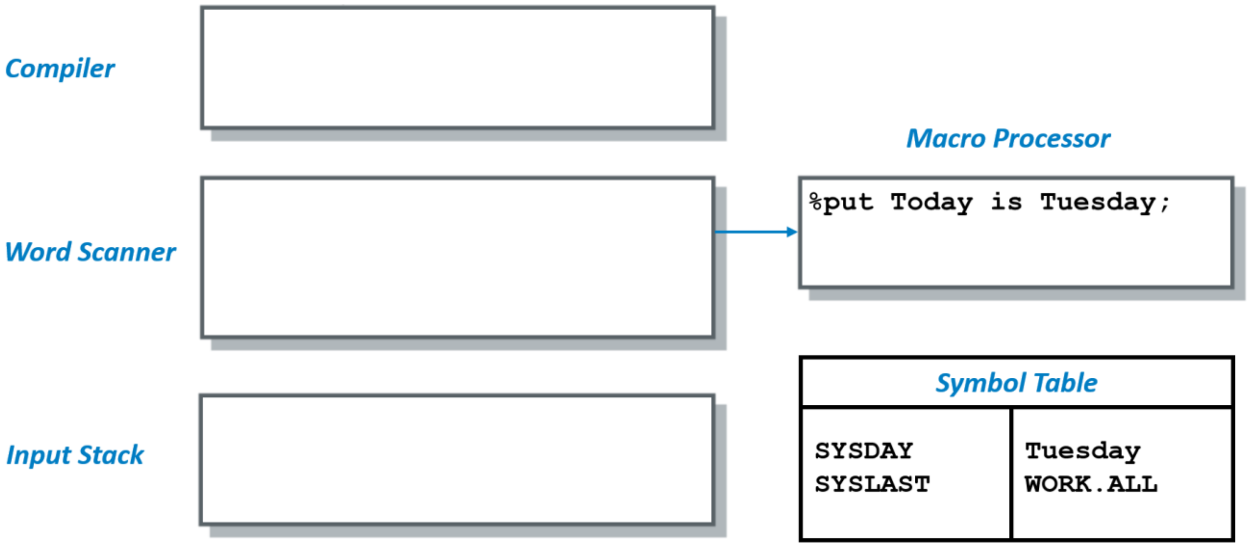
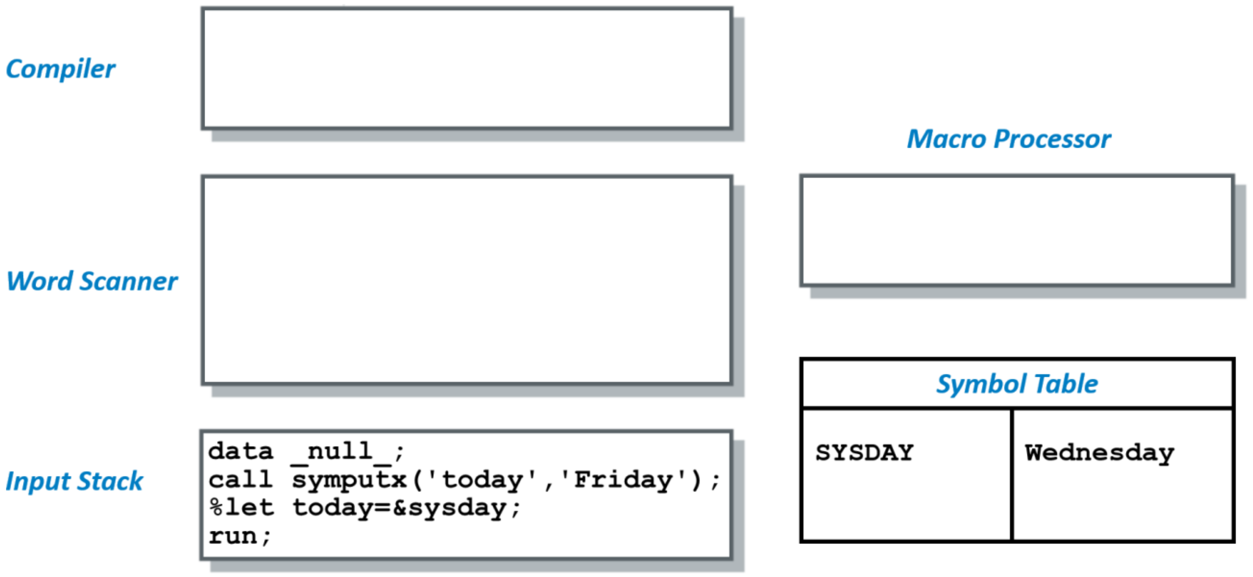


Figure 1.11 Macro Processor Executes Macro Statement

BACK TO THE QUESTION AT THE BEGINNING OF THE PAPER



```
data _null_;
call symputx('today','Friday');
%let today=&sysday;
run;
```

Figure 2.1 SAS Program Submitted to Input Stack

The symputx routine is tokenized into 7 tokens which will be pulled to compiler although the routine is for processing macro variable because it is SAS function not SAS macro language.

You can refer to figure 2.2.

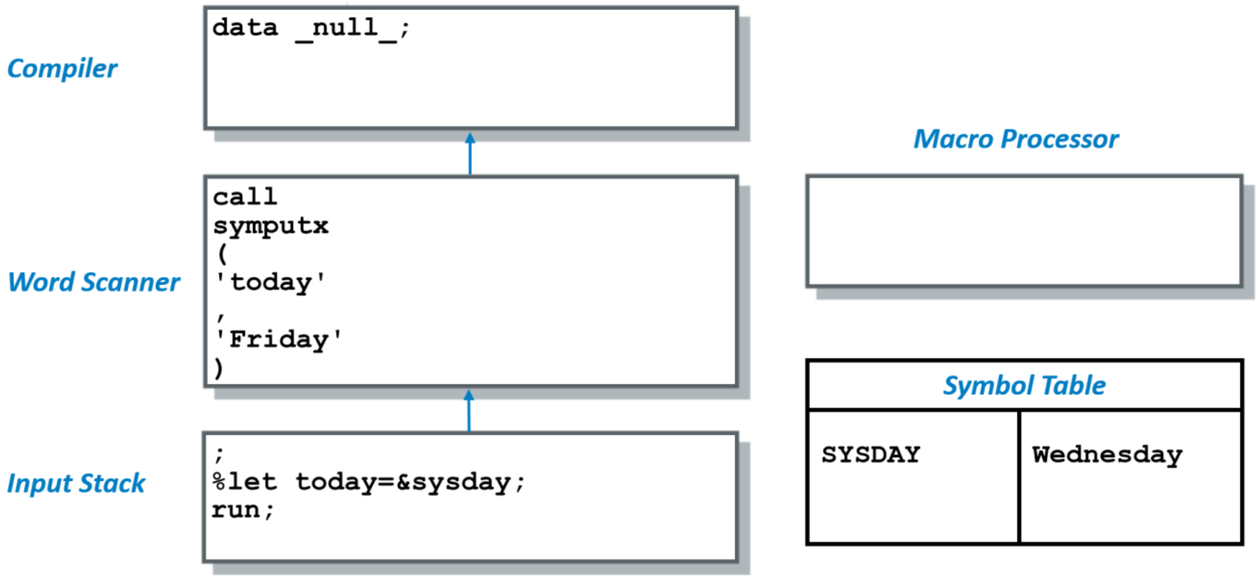


Figure 2.2 Symputx Routine Tokenized into 7 Tokens

Before the boundary is reached, the compiler will not compile and execute the statements. It just remains the statements. And when the macro processor is active, no activity occurs in the word scanner or other compilers.

You can refer to figure 2.3.

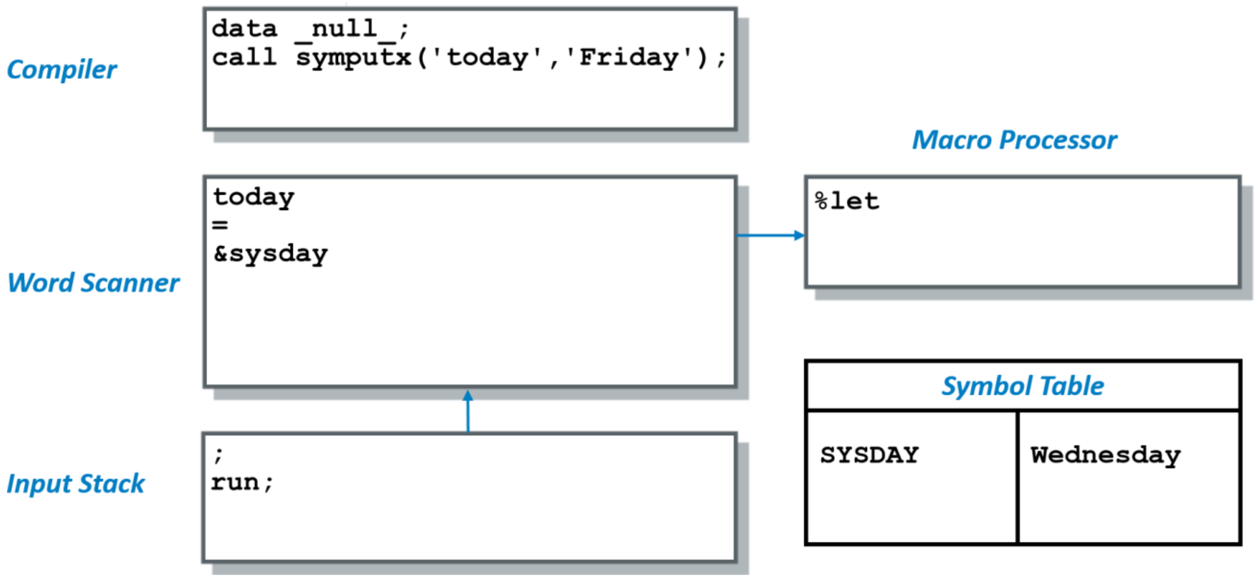


Figure 2.3 Macro Trigger Passed to Macro Processor

%let statement is resolved and executed in macro processor. Macro variable today is generated with value Wednesday.

You can refer to figure 2.4.

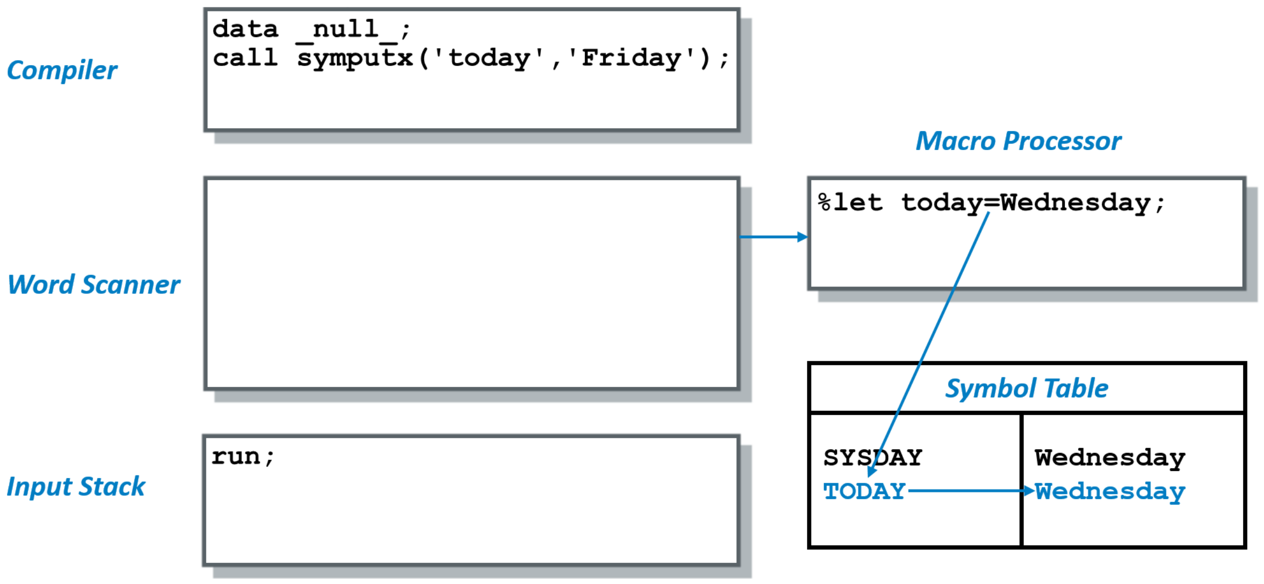


Figure 2.4 Macro Processor Resolves and Executes Macro Statement

When the macro processor action completes, the word scanner continue to read the next token and sends it to the compiler. When a boundary (`run;`) is reached, the word scanner suspends this activity, and the compiler performs a syntax check on the statement.

You can refer to figure 2.5.

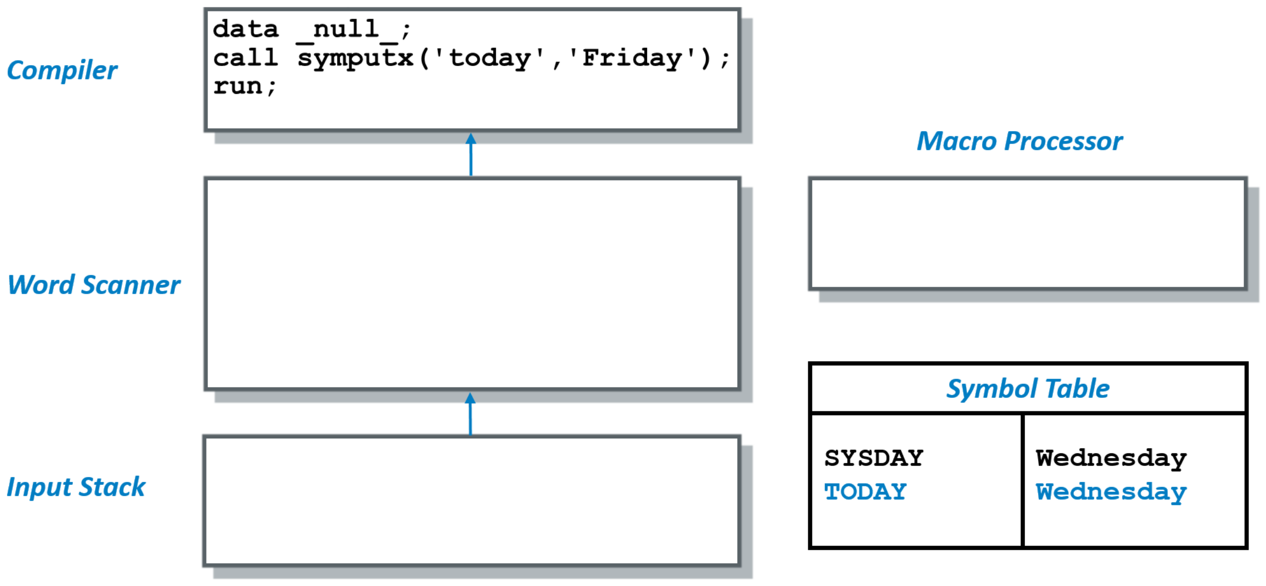


Figure 2.5 Boundary Reached in Compiler

Then compiler executes the DATA step and symputx routine to update the value of macro variable today to Friday.

You can refer to figure 2.6.

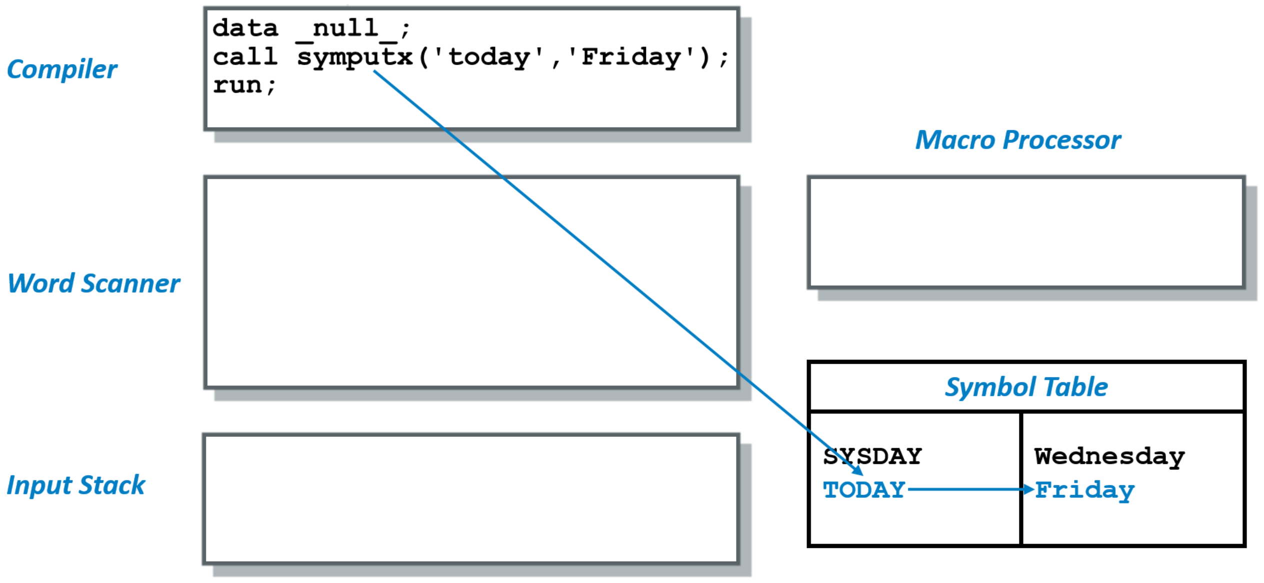


Figure 2.6 DATA Step Executed in Compiler

CONCLUSION

The word scanner reads the SAS program from input stack and then divides it into small chunks called tokens and passes them to the appropriate compiler for eventual execution. That means different SAS language element will be processed by corresponding compiler.

In conclusion, the most important knowledge points are

- All the processing of macro elements is finished in macro processor, including resolution and execution.
- The processing of macro elements is early than other SAS language elements.

To write a SAS program which has non-syntax error as little as possible, you should understand the SAS language processing mechanism. Otherwise, it is easy to get the unexpected result although there is no syntax error.

REFERENCES

- **SAS® Certification Prep Guide – Advanced Programming for SAS®9, Fourth Edition**. Page 296. Copyright © 2014, SAS Institute Inc., Cary, NC, USA
- **SAS Help and Documentation Viewer**. Copyright © 2014, SAS Institute Inc., Cary, NC, USA

CONTACT INFORMATION

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

Name: Roger (Di) Chen
Enterprise: SAS Beijing R&D
Address: 14/F Motorola Plaza, No.1 Wang Jing East Road, Chao Yang District, Beijing, China
City, State ZIP: Beijing, 100102
Work Phone: (8610) 8319 3355-3831
Fax: (8610) 8319 3355 / (8610) 6310 9130
E-mail: di.chen@sas.com
Web:
Twitter: