CALL EXECUTE: A Hidden Treasure and Powerful Function
Rajkumar Sharma, Genentech, South SF, CA.

ABSTRACT

CALL EXECUTE is a very useful DATA step function which helps you stack up many SAS statements and run them together. It interacts with SAS Macro facility for immediate macro execution during the execution of DATA step.

CALL EXECUTE can be used for many tasks that would otherwise need many lines of code, it thus helps to create very compact and efficient SAS Codes.

This paper will explain details about CALL EXECUTE function and how to use it with examples. It also explains difference between standard macro call and call made by CALL EXECUTE.

INTRODUCTION

CALL EXECUTE can be very useful for SAS Programmers who want to perform same task multiple times. It is very useful function which can be used in many different situations for many different purposes. It is also very useful for someone who wants to write compact and efficient piece of code.

CALL EXECUTE enables a program to Interact with the macro facility during the DATA STEP execution. As the work of macro facility takes place before DATA STEP execution begins, information provided by Macro statements have already been processed during the DATA step execution. CALL EXECUTE resolves its argument and executes the resolved value at the next step boundary. If argument resolves to macro Invocation, the macro executes immediately and DATA step execution pauses while the macro executes. If argument resolves to a SAS statement or if execution of the macro generates SAS statements, the statement(s) execute after the end of the DATA step that contains the CALL EXECUTE routine.

SYNTAX:

CALL EXECUTE (argument):

Argument can be one of the following:

- a character string, enclosed in quotation marks. Argument within single quotation marks resolves during program execution. Argument within double quotation marks resolves while the DATA step is being constructed. For example, to invoke the macro NEWDATE, you can use the following code:

  call execute('%newdate');

- the name of a DATA step character variable whose value is a text expression or a SAS statement to be generated. Do not enclose the name
of the DATA step variable in quotation marks. For example, to use the value of the DATA step variable COUNT, which contains a SAS statement or text expression, you can use the following code:

```sas
call execute(count);
```

- a character expression that is resolved by the DATA step to a macro text expression or a SAS statement. For example, to generate a macro invocation whose parameter is the value of the variable MONTH, you use the following code:

```sas
call execute('%newdate'||month||')');
```

**Example 1:**

We will use following TRT dataset to illustrate some examples.

<table>
<thead>
<tr>
<th>Subj</th>
<th>Visit</th>
<th>Trt</th>
<th>Height</th>
<th>Weight</th>
<th>Temp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>WEEK1</td>
<td>POISON</td>
<td>60</td>
<td>180</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
<td>WEEK2</td>
<td>PLACEBO</td>
<td>62</td>
<td>182</td>
</tr>
<tr>
<td>3</td>
<td>102</td>
<td>WEEK3</td>
<td>PLACEBO</td>
<td>61</td>
<td>190</td>
</tr>
<tr>
<td>4</td>
<td>103</td>
<td>WEEK4</td>
<td>PLACEBO</td>
<td>62</td>
<td>195</td>
</tr>
<tr>
<td>5</td>
<td>104</td>
<td>WEEK5</td>
<td>PLACEBO</td>
<td>63</td>
<td>190</td>
</tr>
<tr>
<td>6</td>
<td>105</td>
<td>WEEK6</td>
<td>DRUGA</td>
<td>62</td>
<td>195</td>
</tr>
<tr>
<td>7</td>
<td>106</td>
<td>WEEK7</td>
<td>DRUGA</td>
<td>61</td>
<td>185</td>
</tr>
<tr>
<td>8</td>
<td>107</td>
<td>WEEK8</td>
<td>DRUGA</td>
<td>61</td>
<td>190</td>
</tr>
<tr>
<td>9</td>
<td>108</td>
<td>WEEK9</td>
<td>DRUGA</td>
<td>62</td>
<td>191</td>
</tr>
</tbody>
</table>

In SAS it is not possible to use DATA STEP and PROC Steps together. The following piece of code will not work.

```sas
data trt;
  set trt;
  if trt = 'Placebo' then do;
    proc print data = trt;
  end;
run;
```

SAS will generate error.

**ERROR:** Statement is not valid or it is used out of proper order.

**CALL EXECUTE** can overcome this limitation. The following piece of code will work just fine.
Data trt;
  set trt;
if trt = 'Placebo' then do;
  call execute('Proc print data = trt ; run; ');
end;
run;

In the example above the CALL EXECUTE sends the character string (argument) to the macro facility. The macro facility sees that there is no macro instruction and the argument is a SAS statement. As, the argument is a SAS statement the execution will take place after the data step completes.

Alternatively, if the argument contains macro invocation, then the execution of the macro argument takes place immediately.

Example 2:

CALL SYMPUT is used to create Macro Variable within data step. The macro variable created by CALL SYMPUT can not be referenced inside the creating data step. In case you want to check the value of the newly created macro variable you have to do it with another data step or proc print statement. CALL EXECUTE comes to rescue in this situation.

data trt;
  set libref.trt;
  call symput (Visitn,substr(Visit,5,1));
  call execute('%put the value of visitn is &visitn ; ') ;
run;

Output:

Value of Visitn is 1
Value of Visitn is 2
.
.
Value if Visitn is 9

Example 3:

Many times it’s needed to use same proc steps on different datasets within same directory. In order to do that we need to create many SAS statements or we need to create a macro and pass the name of dataset as the macro argument. CALL EXECUTE is very useful in this situation.

Consider a situation where you want to sort many datasets within same directory. It can be done very easily with CALL EXECUTE and the code is very compact.
data _null_;  
    length dataname $10 ;  
    do datanam = 'dem', 'ae', 'vital', 'edss', 'lab' ;  
        call execute ('proc sort data = libref.'|| trim(left(datanam))|| 'out = ' ||  
                      datanam || '; ' || 'by study subjid visit ; ' || 'run ; ' ) ;  
    run;  

Same thing can be done with Macro but the code will not be as compact.  

%macro doit(datanam=);  
    proc sort data = libref. &datanam ;  
        by study subjid visit ;  
    run;  
%mend doit ;  

%doit(datanam = dem);  
%doit(datanam = ae);  
%doit(datanam = vital);  
%doit(datanam = edss);  
%doit(datanam = lab);  

Example 4:  

Sometime its needed to conditionally run different macro depending on the value  
of a variable. It can be done very efficiently using CALL EXECUTE.  

%macro print;  
    proc print data = libref.trt ;  
    run;  
%mend;  

%macro freq ;  
    proc freq data = libref.trt;  
        tables subjid*trt /list norow nocol nopercent ;  
    run;  
%mend ;  

%macro means ;  
    proc means data = libref.trt ;  
    run;  
%mend;
data _null_;  
    set libref.trt;  
    if height > 60 then call execute('%print ');  
    if height < 60 then call execute('%freq ');  
    if weight > 150 then call execute('%means');  
run;

Depending on the value of the variable call execute will invoke the proper macro.

**Limitations:** CALL EXECUTE does have some limitations. Consider the following case.

```
%macro new;  
    data _null_;  
    set libref.trt;  
    call symput('check',trt);  
    run;  
%put value of check is &check;  
%mend;
```

```
data _null_;  
    call execute('%new');  
run;
```

In this the statement %new will be processed immediately, but the call symput statement with %new macro will not execute until the data statement. Execution of this code will generate warning in the SAS log declaring the macro variable &check not resolved.

**Conclusion:**

CALL EXECUTE can be used effectively to generate compact and efficient SAS code. CALL EXECUTE can combine data steps and proc together and it eliminates the need of typing same piece of code again and again. CALL EXECUTE can be very useful in many different situations but to use it properly and efficiently, one should be aware of how the arguments in CALL EXECUTE are handled by SAS.

**Contact:**

Rajkumar Sharma  
620 E Grand Ave  
South SF, CA 94080  
rajkumas@gene.com
References:

H. Ian Whitlock - CALL EXECUTE: How and Why

Alissa Ruelle and Kitty Moses - CALL EXECUTE: A Primer


SAS is a registered trademark or trademark of SAS Institute Inc. in the USA and other countries.