MAGIC WITH CALL EXECUTE

Bob Virgile
Robert Virgile Associates, Inc.

Overview

CALL EXECUTE lets you stack up SAS® language statements which will run once the current DATA step completes. With a little imagination, this allows you to do the impossible.

Mixing DATA and PROC Steps

You can't mix DATA and PROC steps. Each SAS step must complete before the next step begins. For example, this program is illegal:

```
DATA _NULL_;
SET SALES END=NOMORE;
TOTAL + AMOUNT;
IF NOMORE;
IF TOTAL < 1000000 THEN DO;
   PROC MEANS DATA=SALES;
   CLASS STATE;
END;
ELSE DO;
   PROC MEANS DATA=SALES;
   CLASS STATE YEAR;
END;
```

However, CALL EXECUTE lets you designate the PROC and CLASS statements as statements to be added once the DATA step completes. This approach works, and requires minimal modification to the original:

```
DATA _NULL_;
SET SALES END=NOMORE;
TOTAL + AMOUNT;
IF NOMORE;
IF TOTAL < 1000000 THEN DO;
   CALL EXECUTE
        ('PROC MEANS DATA=SALES;
          CLASS STATE;');
END;
ELSE DO;
   CALL EXECUTE
        ('PROC MEANS DATA=SALES;
          CLASS STATE YEAR;');
END;
RUN;  /* required */
```

To use CALL EXECUTE, two features must be in place. First, the macro processor must be turned on. Secondly, the DATA step must end with a RUN statement. Beyond that, you have great flexibility available. For example, You can use many CALL EXECUTEs in a single DATA step. The code they generate just continues to stack up to be run once the current DATA step completes. The argument to CALL EXECUTE can be a character string (as in the current example), or a more complex expression, such as:

```
CALL EXECUTE (
'PROC PRINT DATA = MOVIES.ROCKY'
|| LEFT(PUT(N,2.)) || ';' );
```

As usual, any variable names not enclosed in quotes get resolved to their values. The result must generate a valid SAS statement.

The argument to CALL EXECUTE can include references to macros or to macro variables. Covering all the variations on that theme would be too lengthy a job for this tip.

Macro Programming in Open Code

You can't use %IF %THEN %ELSE or %DO %END outside of a macro definition. For example, this code would be illegal outside of a macro definition:

```
%IF &CITY = BOSTON %THEN %DO;
    PROC CONTENTS DATA=BOSTON;
%END;
%ELSE %DO;
    PROC PRINT DATA=ANOTHER.CITY;
%END;
```

Instead, you would have to enclose the entire block of code inside of a macro, and then call the macro:

```
%MACRO MUSTUSE;

   %IF &CITY = BOSTON %THEN %DO;
       PROC CONTENTS DATA=BOSTON;
   %END;
   %ELSE %DO;
       PROC PRINT DATA=ANOTHER.CITY;
   %END;

%MEND MUSTUSE;

%MUSEUSE
```

Once again, CALL EXECUTE comes to the rescue, letting you replace a macro comparison with a DATA step comparison:

```
DATA _NULL_;
IF LEFT("&CITY") = "BOSTON"
THEN CALL EXECUTE (
'PROC CONTENTS DATA=BOSTON;' );
ELSE CALL EXECUTE (
'PROC PRINT DATA=ANOTHER.CITY;' );
RUN;
```

I expect this to be a temporary work-around, which handles just some of the macro %IF %THEN comparisons. At some point in the future, I expect that macro programming statements will be permitted in open code.

In similar fashion, %DO %END are not permitted in open code. CALL EXECUTE can eliminate the need, at the same time simplifying the program. This program attempts to print the first 10 observations from every SAS data set in the library:

```
PROC CONTENTS DATA=LIB._ALL_
   NOPRINT OUT=TEMP (KEEP=MEMNAME);

PROC SORT DATA=TEMP NODUPKEY;
BY MEMNAME;

DATA _NULL_;
SET TEMP END=NOMORE;
CALL SYMPUT
('D' || COMPRESS(PUT(_N_,3.)),
 TRIM(MEMNAME));
IF NOMORE THAN CALL SYMPUT
('TOTDS', PUT(_N_,3.));
RUN;

%DO I=1 %TO &TOTDS;

 PROC PRINT DATA=LIB.&&D&I (OBS=10);
 TITLE "*** &&D&I ***";
 RUN;

%END;
```

As was the case with %IF %THEN, this program would need to define a macro, just to be able to execute the %DO loop. Instead, CALL EXECUTE comes to the rescue. After PROC SORT, the DATA step and the %DO loop can be replaced by a single DATA step:

```
DATA _NULL_;
SET TEMP;
CALL EXECUTE ('PROC PRINT DATA='
|| MEMNAME ||
'(OBS=10); TITLE "*** ' ||
TRIM(MEMNAME) || ' ***"; RUN;');
RUN;
```

Even if %DO loops are eventually permitted in open code, CALL EXECUTE may be the simpler solution.

Contact information:

Robert Virgile
Robert Virgile Associates, Inc.
3 Rock Street
Woburn, MA 01801
(617) 938-0307

SAS is a registered trademark of SAS Institute, Inc.